Computational construction of W-graphs associated with Hecke algebras(Computational Geometry and Discrete Geometry)

AUTHOR(S):

落合, 豊行

数理解析研究所講究録
第872巻 1994年 108-143

Computational construction of W-graphs associated with Hecke algebras

奈良女子大学
理学部情報科学科　落合　豊行
Mitsuyuki Ochiai

1。序論

筆者は、１９８７年以来結び目理論研究支援用ソフトウェア、KnotTheorybyComputer, を研究開発し、国内外で研究発表し、２年位前から国内外にKnotTheorybyComputerを配付してきた。講演をする毎にこのソフトウェアは、機能を拡張し成長してきた。しかしながら、現在のバージョンでも、次に述べるような不満がある。

（１）２次元グラフィクスによる、結び目の正則射影（結び目理論を研究するにはこの方法を用いるの現在の処が最適である）を、３次元グラフィクスによるより立体的な表示機能を与えること。

（２）より交点数の多い結び目（例えば、２００交点位まで、または１６ブレイド位まで）の多項式不変量を高速に計算すること。

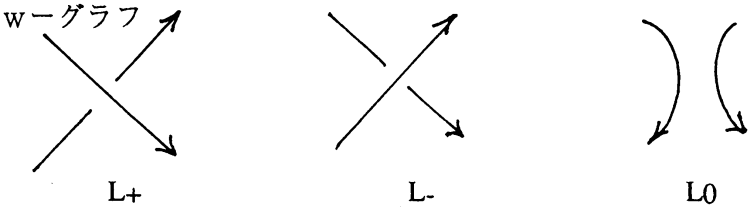
（３）マウスによる結び目のイソトピー変形を完璧に行なうこと。

（１）の機能に関して、Geometry Center （Minnesota University ）が開発した geomview は極めて優れたソフトウェアである、しかし geomview は visualization のみに特化したソフトウェアで結び目のイソトピー変形とか、多項式不変量を計算する機能を全く持たない。

今年３月に、筆者は同センタで開催されたKnot Workshop に参加し、Geometry Center に１箇月滞在し、同センタが開発した visualization に関するソフトウェア技術を修得することに努めた。

Knot Workshop では、筆者は、” Computer aided Knot theory with OSF/Motif ” という題目で、ワークステーション上での KnotTheorybyComputer についてのデモ講演をすると同時に,（２）に関して最近開発したコンピュータプログラムの紹介をした。ここでは、その方法について解説し、巻末に７、８ブレイドに対応したWーグラフを掲載する。

2。wーグラフ



(1) $P_T(t,x) = 1$ 、Tは自明な結び目

(2) $t^{-1}P_{L+}(t,x) - tP_{L-}(t,x) = xP_{L0}(t,x)$

図ー１　Conway 関係式による２変数Jones polynomial

現在知られている結び目の多項式不変量は、全て結び目の１つの交点における３状態（プラス交点L+、マイナス交点L-、平滑化L0（または、逆平滑化を含めた４状態））から派生する二分木（４状態の場合には、３分木）を用いて計算可能である。すなわち、Conway 関係式（図－１）から計算出来る。しかし、この関係式を用いた場合、計算回数は結び目の交点数の冪乗のオーダで増加する。従って、極めて多くの交点を持つ結び目の多項式不変量を求めるにはもっと別の巧妙な方法が必要である。

１つの方法として、結び目のブレイド表示から行列への表現が有効である。

n次ブレイド群 B_n とは、次のような群表示を持つ群である。

$$B_n = < \sigma 1, \sigma 2, \ldots, \sigma n\text{-}1 \mid \sigma i \sigma j = \sigma j \sigma i \ (\ |i-j| > 1),$$
$$\sigma i \sigma i+1 \sigma i = \sigma i+1 \sigma i \sigma i+1 \ (i = 1,2,\ldots,n-2) >$$

$$X_{L(q,\lambda)} = (-(1-\lambda q)/(\sqrt{\lambda}(1-q))^{n-1}\sqrt{\lambda}^{e}\ \mathrm{trace}(\pi(b))$$

e は b の符号和で、π は B_n から Hecke algebra H(q,n) への表現である。

このとき、任意の結び目Ｌは、あるｎ次ブレイド B_n の元ｂの閉包ｂ^としてかけ、Ｌの２変数Jones 多項式Ｐ(L;t,x) は、B_n の行列表現のトレースの１次結合 $X_{L(q,\lambda)}$ としてかける。このとき、$t = \sqrt{\lambda}\sqrt{q}$、$x = (\sqrt{q} - 1/\sqrt{q})$ である[4]。

問題は、表現行列を具体的にどう求めるかである。

ここでは、Lascoux-Schuzenberger により考案された Young 図形からＷ－グラフを構成し、Ｗ－グラフから表現行列を作成する方法を採用する。この方法は、Lascoux から京都大学の行者氏への私信で伝えられたもので[1, 2]、厳密な数学的証明はまだ与えられていないようである。しかし、作成した表現行列の妥当性は、B_n の関係式を満たすかどうか調べることで確認できる。

Lascoux-Schuzenberger によれば、自然数ｎに対してH(q,n)（または、B_n）の行列表現は次のように求められる：ｎ＝６の場合について説明する。

Λ(n) をｎの分割数、すなわちｎをいくつかの非負の整数の和で表す表し方の数とする。この場合、Λ(6) = 11 であり、具体的にかくと｛｛6｝、｛5、1｝、｛4、2｝、｛4、1、1｝、｛3、3｝、｛3、2、1｝、｛3、1、1，1｝、｛2、2、2｝、｛2、2、1、1｝、｛2、1、1、1、1｝、｛1、1、1、1、1、1｝｝となる。

このとき、各分割に対して台を用意する、例えば、｛4、1、1｝に対しては
を、｛3、3｝に対して　　を用意する。

次に、例えば｛3、2、1｝（この場合、深さ３）に対して用意した台に、数字を左から右に、上から下に数字が増加するように箱に割り振る。例えば、次のようになる。

123	124	134	125	135	123	124	134	125	135	145	126	136
45	35	25	34	24	46	36	26	36	26	26	34	24
6	6	6	6	6	5	5	5	4	4	3	5	5

1 2 6　1 3 6　1 4 6
3 5　　2 5　　2 5
4　　4　　3

これらの１６個の台の割当を Young 図形の標準盤と言う。次に、各標準盤に割り当てられた数字を、　　の順序で数字を並べると、

x_1=325146, x_2=425136, x_3=435126, x_4=524136, x_5=534126, x_6=326145, x_7=426135, x_8=436125, x_9=526134, x_{10}=536124,x_{11}=546123, x_{12}=624135, x_{13}=634125, x_{14}=625134, x_{15}=635124, x_{16}=645123

という word が得られる。

これらの word の１つ１つをW-グラフＧの頂点$x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}, x_{16}$として考える。これらの内、word w、w’につき、wの内の２文字 i, j (i < j) を入れ替えると、w’になり、i と j との間には、i < m < j を満たすmが存在しなければ、wとw’との間に辺があるとする。例えば、325146 と 425136 と、325146 と 326145 との間には辺がある。また、425136 と 625134 とでは、４と６とが入れ替わっているが、５が間に挟まっているのでこの段階では辺で結ばない。

次に、３つの続いた数字、例えば 2, 3, 4 に着目する。この時、word w に対して wP を次のように定義する。

w＝...2...3...4...　----->　wP は定義されない
w＝...2...4...3...　----->　wP = ...3...4...2...
w＝...3...2...4...　----->　wP = ...4...2...3...
w＝...3...4...2...　----->　wP = ...2...4...3...
w＝...4...2...3...　----->　wP = ...3...2...4...
w＝...4...3...2...　----->　wP は定義されない

つまり、真中に、234 のうちの真中の数３がくると、wP は定義されない。そうでなければ、両端を入れ替える。ここで、w_1 と w_2 が既に辺で結ばれていたら、w_1P と w_2P との間に辺を設ける。例えば、w_1 = 325146 , w_2 = 534126 とすれば、これらは辺で結ばれているので、w_1P = 425136 と w_2P = 524136 との間に辺を設ける。

2, 3, 4 に限らず、1, 2, 3 にも 3, 4, 5 にも 4, 5, 6 にも同じことを繰り返して可能な限り辺を付け加える。

更に、このようにして得られたw_1P と w_2P に関しても（w_1P）P と（w_2P）P を作ることにより辺を付け加える。この操作に関しても可能な限り行う。

上のW－グラフの隣接行列 A を書くと、次のようになる：

0100210020000000
1011001000000000
0100100100000000
0100100010210000
2011000001001000
1000001000002000
0100010110010000
0010001001001002
2001001001000100
0000100110100010
0002000001000001
0001001000001102
0000120100010010
0000000010010010
0000000001001101
0000000200120010

左の隣接行列で1は最初の辺の条件から生成される辺を、1より大きい数はそれ以降の生成条件から出来る辺を表す

3。結び目不変量の行列表現

3。1 行列表現

次に、wーグラフからBnの行列表現を得るための方法について考える。

まず、wーグラフGの各頂点xに対して、I(x)をi∈I(={1,2,3,4,5})でi+1がiを含む行より下にあるものの集合として定義する。例えば、$I(x_1)=\{3,5\}$である。また、IC(x)をIにおけるI(x)の補集合とする。すなわち、$IC(x_1)=\{1,2,4\}$である。この時、σ_jに対応する表現行列T_jは、16次の正方行列で、対角成分(i, i)は、$j \in I(x_i)$の時、-1で、$j \notin I(x_i)$の時tである。それ以外の成分(l, m)($l \neq m$)については、$j \in I(x_l)$かつ$j \notin I(x_m)$かつA(l,m)が0でない時にのみ$\sqrt{q}$で、それ以外は0である。

3。2 B_7、B_8のwーグラフ

wーグラフの頂点xは、I(x)で完全に識別できるので、xをそのI(x)でラベル付け、xをI(x)で表わす。例えば、x_1=325146, x_2=425136,..., x_{15}=635124, x_{16}=645123を、x_1=[3 5], x_2=[2 4 5],..., x_{15}=[1 3], x_{16}=[1 2 4]の様に表わす。

Matrix size = 14, reduced Matrix size = 1

0 1 1 0 2 1 0 4 2 1 3 1 2 3 1 2 5 1 3 4 1 3 6 1 3 8 2 4 7 1 5 6 1 5 9 1
6 7 1 6 10 1 7 8 1 7 11 1 7 13 2 8 12 1 9 10 1 10 11 1 11 12 1 12 13 1

上のデータは、wーグラフの隣接行列を表わす。3つ組(a, b, c)、例えば0 1 1はx_1とx_2とが辺で結ばれていることを示す。3番目の1は最初の隣接関係で生成された辺であることを示す。0 4 2の場合には、x_1とx_5が2番目の隣接関係で生成された辺であることを示す。

B7のｗーグラフ　　B8のｗーグラフ
図ー２、　　図ー３

B_6のｗーグラフは、[6] に言及されているように、Gyoja-Naruse により既に作られている（ただし、Young 図形の並べ方が[4] によるものと相対になっているので I(x) が IC(x) となっている）。ここでは、B_7、B_8のｗーグラフのみを掲載する（図ー２、３）。

$B_9, B_{10}, B_{11}, B_{12}$のｗーグラフのデータは、次のようにあまりに膨大なので割愛する。

B9　　105452 バイト、　　B10　　594020 バイト、
B11　　2854938 バイト、　　B12　　11318332 バイト

巻末に掲載したプログラムによれば、B_{16}までのｗーグラフを作成することが出来る[7]。それ以上のものについては記憶領域を使用する上での特別の配慮が必要とされる。

実は、Lascoux-Schuzenbergerの方法を用いて２年前に大学院の学生が、紙とエンピツで、深さ２までのYoung 図形（これは、１変数Jones 多項式）に対応するｗーグラフを１２ブレイドまで、すべての深さのYoung 図形（これは、２変数Jones 多項式）に対応するｗーグラフを８ブレイドまで計算した[11]。彼女の計算は、１変数Jones 多項式の１１ブレイドまでは正しいが、残念ながら１２ブレイドの計算に誤りがあった。今年の初めに彼女の計算結果を,KnotTheorybyComputer に移植する際に,始めてその誤りにきずいたときには、筆者も含めて誰もその計算を復元することは不可能であった。

そこで、Lascoux-Schuzenberger の方法を直接プログラムすることにした。最初は比較的簡単な１変数Jones 多項式についてのプログラムの制作から始め、それを２週間位で完成出来た。次に２変数Jones 多項式ついても２箇月位で完成出来た。実際に使用可能な形のプログラムは、アメリカでの講演が済んで日本に帰国してからである。

また、１２ブレイドのあたりの２変数Jones 多項式（実際には、４ブレイドの３ーparallel link に関係する１変数Jones 多項式）を求めるには、１変数多項式をエントリに持つ約 8000 次の正方行列の行列計算が必要となるが、これは大型疎行列を１次元配列を効率よく用いて計算するプログラムを開発することにより可能となった。

ところで、mutant な２つの結び目を一般的に判別する多項式 不変量は、現在でもまだ知られていない。筆者は大阪大学の村上順氏の協力を得て、１２ブレイドのｗーグラフの部分グラフから、表現行列の部分表現行列を作り、これからmutant な２つの結び目を判別する多項式 不変量を作った。この多項式 不変量は、寺坂ー樹下結び目と Conway 結び目を判別出来る[8]。

巻末に制作したプログラムのリストを掲載したので興味ある読者は利用してください。

参考文献

[1] A. Gyoja, Topological Invariants of Links and Representations of Hecke Algebras, preprint.

[2] , Topological Invariants of Links and Representations of Hecke Algebras II, preprint.

[3] V.F.R.Jones, A polynomial invariants for knots via von Neumann algebras, Bull.A.M.S.,12(1985), 103-111.

[4] V.F.R.Jones, Hecke algebra representations of braid groups and link polynomias, Annals of

Mathematics, 126(1987), 335-388.

[5] A.Lascoux and M.P.Schutzenberger, Polynomes de Kazhdan-Lusztig pour les grammanniennes, Asterisque, (1981),87-88,249-266.

[6] J. Murakami, The Parallel Version of Polynomial Invariants of Links, Osaka J. 26 (1989),1-55.

[7] M. Ochiai, Computational construction of W-graphs associated with Hecke algebras Hn for n up to 12, preprint.

[8] M. Ochiai and J.Murakami, Subgraphs of W-graphs and 3-parallel invariants of knots, preprint.

[9] 河内明夫編、結び目理論、Springer-Verlag 1990.

[10] 村上　順、組紐群と絡み目の不変量、preprint.

[11] 吉田暁美、１変数、２変数 Jones 多項式の W-graph による Burau 表現の具体的構成、１９９２年修士論文。

W－graph for B7

partition: 7 0 0 0 0 0 0 0
x1 = [1 2 3 4 5 6]
partition: 6 1 0 0 0 0 0 0
x1 = [2 3 4 5 6] x2 = [1 3 4 5 6] x3 = [1 2 4 5 6] x4 = [1 2 3 5 6] x5 = [1 2 3 4 6] x6 = [1 2 3 4 5]
partition: 5 2 0 0 0 0 0 0
x1 = [2 4 5 6] x2 = [1 3 4 5 6] x3 = [2 3 5 6] x4 = [1 3 5 6] x5 = [1 2 4 5 6]
x6 = [2 3 4 6] x7 = [1 3 4 6] x8 = [1 2 4 6] x9 = [1 2 3 5 6] x10 = [2 3 4 5]
x11 = [1 3 4 5] x12 = [1 2 4 5] x13 = [1 2 3 5] x14 = [1 2 3 4 6]
partition: 5 1 1 0 0 0 0 0
x1 = [3 4 5 6] x2 = [2 4 5 6] x3 = [1 4 5 6] x4 = [2 3 5 6] x5 = [1 3 5 6]
x6 = [1 2 5 6] x7 = [2 3 4 6] x8 = [1 3 4 6] x9 = [1 2 4 6] x10 = [1 2 3 6]
x11 = [2 3 4 5] x12 = [1 3 4 5] x13 = [1 2 4 5] x14 = [1 2 3 5] x15 = [1 2 3 4]
partition: 4 3 0 0 0 0 0 0
x1 = [2 4 6] x2 = [1 3 4 6] x3 = [2 3 5 6] x4 = [1 3 5 6] x5 = [1 2 4 5 6]
x6 = [2 4 5] x7 = [1 3 4 5] x8 = [2 3 5] x9 = [1 3 5] x10 = [1 2 4 5]
x11 = [2 3 4 6] x12 = [1 3 4 6] x13 = [1 2 4 6] x14 = [1 2 3 5 6]
partition: 4 2 1 0 0 0 0 0
x1 = [3 5 6] x2 = [2 4 5 6] x3 = [1 4 5 6] x4 = [2 5 6] x5 = [1 3 5 6]
x6 = [3 4 6] x7 = [2 4 6] x8 = [1 4 6] x9 = [2 3 5 6] x10 = [1 3 5 6]
x11 = [1 2 5 6] x12 = [2 4 6] x13 = [1 3 4 6] x14 = [2 3 6] x15 = [1 3 6]
x16 = [1 2 4 6] x17 = [3 4 5] x18 = [2 4 5] x19 = [1 4 5] x20 = [2 3 5]
x21 = [1 3 5] x22 = [1 2 5] x23 = [2 3 4 6] x24 = [1 3 4 6] x25 = [1 2 4 6]
x26 = [1 2 3 6] x27 = [2 4 5] x28 = [1 3 4 5] x29 = [2 3 5] x30 = [1 3 5]
x31 = [1 2 4 5] x32 = [2 3 4] x33 = [1 3 4] x34 = [1 2 4] x35 = [1 2 3 5]
partition: 4 1 1 1 0 0 0 0
x1 = [4 5 6] x2 = [3 5 6] x3 = [2 5 6] x4 = [1 5 6] x5 = [3 4 6]
x6 = [2 4 6] x7 = [1 4 6] x8 = [2 3 6] x9 = [1 3 6] x10 = [1 2 6]
x11 = [3 4 5] x12 = [2 4 5] x13 = [1 4 5] x14 = [2 3 5] x15 = [1 3 5]
x16 = [1 2 5] x17 = [2 3 4] x18 = [1 3 4] x19 = [1 2 4] x20 = [1 2 3]
partition: 3 3 1 0 0 0 0 0
x1 = [3 5] x2 = [2 4 5] x3 = [1 4 5] x4 = [2 5] x5 = [1 3 5]
x6 = [3 4 6] x7 = [2 4 6] x8 = [1 4 6] x9 = [2 3 5 6] x10 = [1 3 5 6]
x11 = [1 2 5 6] x12 = [2 4 6] x13 = [1 3 4 6] x14 = [2 3 6] x15 = [1 3 6]
x16 = [1 2 4 6] x17 = [2 4] x18 = [1 3 4] x19 = [2 3 5] x20 = [1 3 5] x21 = [1 2 4 5]
partition: 3 2 2 0 0 0 0 0
x1 = [3 6] x2 = [2 4 6] x3 = [1 4 6] x4 = [2 5 6] x5 = [1 3 5 6]
x6 = [3 5] x7 = [2 4 5] x8 = [1 4 5] x9 = [2 5] x10 = [1 3 5]
x11 = [3 4] x12 = [2 4] x13 = [1 4] x14 = [2 3 5] x15 = [1 3 5]
x16 = [1 2 5] x17 = [2 4 6] x18 = [1 3 4 6] x19 = [2 3 6] x20 = [1 3 6] x21 = [1 2 4 6]
partition: 3 2 1 1 0 0 0 0
x1 = [4 6] x2 = [3 5 6] x3 = [2 5 6] x4 = [1 5 6] x5 = [3 6]
x6 = [2 4 6] x7 = [1 4 6] x8 = [2 6] x9 = [1 3 6] x10 = [4 5]
x11 = [3 5] x12 = [2 5] x13 = [1 5] x14 = [3 4 6] x15 = [2 4 6]
x16 = [1 4 6] x17 = [2 3 6] x18 = [1 3 6] x19 = [1 2 6] x20 = [3 5]
x21 = [2 4 5] x22 = [1 4 5] x23 = [2 5] x24 = [1 3 5] x25 = [3 4]

x26 = [2 4] x27 = [1 4] x28 = [2 3 5] x29 = [1 3 5] x30 = [1 2 5]
x31 = [2 4] x32 = [1 3 4] x33 = [2 3] x34 = [1 3] x35 = [1 2 4]
partition: 3 1 1 1 1 0 0 0
x1 = [5 6] x2 = [4 6] x3 = [3 6] x4 = [2 6] x5 = [1 6]
x6 = [4 5] x7 = [3 5] x8 = [2 5] x9 = [1 5] x10 = [3 4]
x11 = [2 4] x12 = [1 4] x13 = [2 3] x14 = [1 3] x15 = [1 2]
partition: 2 2 2 1 0 0 0 0
x1 = [4] x2 = [3 5] x3 = [2 5] x4 = [1 5] x5 = [3 6]
x6 = [2 4 6] x7 = [1 4 6] x8 = [2 6] x9 = [1 3 6] x10 = [3]
x11 = [2 4] x12 = [1 4] x13 = [2 5] x14 = [1 3 5]
partition: 2 2 1 1 1 0 0 0
x1 = [5] x2 = [4 6] x3 = [3 6] x4 = [2 6] x5 = [1 6]
x6 = [4] x7 = [3 5] x8 = [2 5] x9 = [1 5] x10 = [3]
x11 = [2 4] x12 = [1 4] x13 = [2] x14 = [1 3]
partition: 2 1 1 1 1 1 0 0
x1 = [6] x2 = [5] x3 = [4] x4 = [3] x5 = [2] x6 = [1]
partition: 1 1 1 1 1 1 1 0
Matrix size = 6, reduced Matrix size = 0
0 1 1 1 2 1 2 3 1 3 4 1 4 5 1
Matrix size = 14, reduced Matrix size = 1
0 1 1 0 2 1 0 4 2 1 3 1 2 3 1 2 5 1 3 4 1 3 6 1 3 8 2 4 7 1 5 6 1 5 9 1
6 7 1 6 10 1 7 8 1 7 11 1 7 13 2 8 12 1 9 10 1 10 11 1 11 12 1 12 13 1
Matrix size = 15, reduced Matrix size = 1
0 1 1 1 2 1 1 3 1 2 4 1 3 4 1 3 6 1 4 5 1 4 7 1 5 8 1 6 7 1 6 10 1
7 8 1 7 11 1 8 9 1 8 12 1 9 13 1 10 11 1 11 12 1 12 13 1 13 14 1
Matrix size = 14, reduced Matrix size = 2
0 1 1 0 2 1 0 4 2 0 5 1 0 10 2 0 12 3 1 3 1 1 6 1 1 11 2 2 3 1 2 7 1 2 13 4
3 4 1 3 8 1 4 9 1 5 6 1 5 7 1 5 9 2 6 8 1 7 8 1 7 10 1
8 9 1 8 11 1 8 13 2 9 12 1 10 11 1 11 12 1 12 13 1
Matrix size = 35, reduced Matrix size = 4
0 1 1 0 4 2 0 5 1 0 8 2 1 2 1 1 3 1 1 6 1 2 4 1 2 7 1 3 4 1 3 8 1 3 10 2 3 11 1
4 9 1 4 12 1 5 6 1 5 12 2 5 16 1 6 7 1 6 8 1 6 11 1 6 17 1 6 22 2
7 9 1 7 12 1 7 15 2 7 18 1 7 23 2 8 9 1 8 13 1 8 19 1 9 10 1 9 14 1 9 20 1
10 15 1 10 21 1 11 12 1 11 13 1 11 15 2 11 26 1 12 14 1 12 27 1 13 14 1 13 22 1 13 28 1
14 15 1 14 23 1 14 25 2 14 29 1 15 24 1 15 30 1 16 17 1 16 27 2 17 18 1 17 19 1 17 26 1
18 20 1 18 27 1 18 30 2 19 20 1 19 22 1 19 28 1 20 21 1 20 23 1 20 29 1
21 24 1 21 30 1 21 34 2 22 23 1 22 31 1 23 24 1 23 32 1 24 25 1 24 33 1 25 34 1
26 27 1 26 28 1 26 30 2 27 29 1 28 29 1 28 31 1 29 30 1 29 32 1 29 34 2
30 33 1 31 32 1 32 33 1 33 34 1
Matrix size = 20, reduced Matrix size = 2
0 1 1 1 2 1 1 4 1 2 3 1 2 5 1 3 6 1 4 5 1 4 10 1 5 6 1 5 7 1 5 11 1
6 8 1 6 12 1 7 8 1 7 13 1 8 9 1 8 14 1 9 15 1 10 11 1 11 12 1 11 13 1
12 14 1 13 14 1 13 16 1 14 15 1 14 17 1 15 18 1 16 17 1 17 18 1 18 19 1
Matrix size = 21, reduced Matrix size = 3
0 1 1 0 4 2 0 5 1 0 8 2 1 2 1 1 3 1 1 6 1 2 4 1 2 7 1 2 20 3
3 4 1 3 8 1 3 10 2 3 11 1 3 18 2 3 20 3 4 9 1 4 12 1 4 19 2 5 6 1 5 12 2
6 7 1 6 8 1 6 11 1 7 9 1 7 12 1 7 15 2 8 9 1 8 13 1 9 10 1 9 14 1 10 15 1

11 12 1 11 13 1 11 15 2 11 16 1 12 14 1 12 17 1 13 14 1 13 18 1 14 15 1 14 19 1
15 20 1 16 17 1 16 18 1 16 20 2 17 19 1 18 19 1 19 20 1
Matrix size = 21, reduced Matrix size = 3
0 1 1 0 4 2 0 5 1 0 17 3 0 18 3 1 2 1 1 3 1 1 6 1 1 16 2 2 4 1 2 7 1 2 17 2
3 4 1 3 8 1 4 9 1 5 6 1 5 9 2 5 10 1 5 13 2 6 7 1 6 8 1 6 11 1 7 9 1 7 12 1
8 9 1 8 13 1 8 15 2 8 16 1 9 14 1 9 17 1 10 11 1 10 17 2 11 12 1 11 13 1 11 16 1
12 14 1 12 17 1 12 20 2 13 14 1 13 18 1 14 15 1 14 19 1 15 20 1 16 17 1 16 18 1 16 20 2
17 19 1 18 19 1 19 20 1
Matrix size = 35, reduced Matrix size = 4
0 1 1 0 5 2 0 9 1 0 13 2 1 2 1 1 4 1 1 10 1 2 3 1 2 5 1 2 11 1 3 6 1 3 12 1
4 5 1 4 8 2 4 13 1 4 16 2 4 19 1 5 6 1 5 7 1 5 14 1 5 20 1 6 8 1 6 15 1 6 21 1
7 8 1 7 16 1 7 18 2 7 22 1 8 17 1 8 23 1 9 10 1 9 20 2 10 11 1 10 13 1 10 19 1
11 12 1 11 14 1 11 20 1 11 27 2 12 15 1 12 21 1 12 28 2 13 14 1 13 24 1
14 15 1 14 16 1 14 25 1 15 17 1 15 26 1 16 17 1 16 27 1 17 18 1 17 28 1 18 29 1
19 20 1 19 23 2 19 24 1 19 27 2 20 21 1 20 22 1 20 25 1 21 23 1 21 26 1
22 23 1 22 27 1 22 29 2 22 30 1 23 28 1 23 31 1 24 25 1 24 31 2 25 26 1 25 27 1 25 30 1
26 28 1 26 31 1 26 34 2 27 28 1 27 32 1 28 29 1 28 33 1 29 34 1 30 31 1 30 32 1 30 34 2
31 33 1 32 33 1 33 34 1
Matrix size = 15, reduced Matrix size = 1
0 1 1 1 2 1 1 5 1 2 3 1 2 6 1 3 4 1 3 7 1 4 8 1 5 6 1 6 7 1 6 9 1
7 8 1 7 10 1 8 11 1 9 10 1 10 11 1 10 12 1 11 13 1 12 13 1 13 14 1
Matrix size = 14, reduced Matrix size = 2
0 1 1 0 5 2 0 11 4 1 2 1 1 4 1 1 13 3 2 3 1 2 5 1 2 12 2 3 6 1 3 13 2
4 5 1 4 8 2 4 9 1 5 6 1 5 7 1 5 10 1 6 8 1 6 11 1 7 8 1 7 12 1 8 13 1
9 10 1 9 13 2 10 11 1 10 12 1 11 13 1 12 13 1
Matrix size = 14, reduced Matrix size = 1
0 1 1 0 6 2 1 2 1 1 5 1 2 3 1 2 6 1 3 4 1 3 7 1 4 8 1 5 6 1 5 10 2
6 7 1 6 9 1 7 8 1 7 10 1 8 11 1 9 10 1 9 13 2 10 11 1 10 12 1 11 13 1 12 13 1
Matrix size = 6, reduced Matrix size = 0
0 1 1 1 2 1 2 3 1 3 4 1 4 5 1

W−graph for B8

partition: 8 0 0 0 0 0 0 0 0

x1 = [1 2 3 4 5 6 7]

partition: 7 1 0 0 0 0 0 0 0

x1 = [2 3 4 5 6 7] x2 = [1 3 4 5 6 7] x3 = [1 2 4 5 6 7] x4 = [1 2 3 5 6 7] x5 = [1 2 3 4 6 7]

x6 = [1 2 3 4 5 7] x7 = [1 2 3 4 5 6]

partition: 6 2 0 0 0 0 0 0 0

x1 = [2 4 5 6 7] x2 = [1 3 4 5 6 7] x3 = [2 3 5 6 7] x4 = [1 3 5 6 7] x5 = [1 2 4 5 6 7]

x6 = [2 3 4 6 7] x7 = [1 3 4 6 7] x8 = [1 2 4 6 7] x9 = [1 2 3 5 6 7] x10 = [2 3 4 5 7]

x11 = [1 3 4 5 7] x12 = [1 2 4 5 7] x13 = [1 2 3 5 7] x14 = [1 2 3 4 6 7] x15 = [2 3 4 5 6]

x16 = [1 3 4 5 6] x17 = [1 2 4 5 6] x18 = [1 2 3 5 6] x19 = [1 2 3 4 6] x20 = [1 2 3 4 5 7]

partition: 6 1 1 0 0 0 0 0 0

x1 = [3 4 5 6 7] x2 = [2 4 5 6 7] x3 = [1 4 5 6 7] x4 = [2 3 5 6 7] x5 = [1 3 5 6 7]

x6 = [1 2 5 6 7] x7 = [2 3 4 6 7] x8 = [1 3 4 6 7] x9 = [1 2 4 6 7] x10 = [1 2 3 6 7]

x11 = [2 3 4 5 7] x12 = [1 3 4 5 7] x13 = [1 2 4 5 7] x14 = [1 2 3 5 7] x15 = [1 2 3 4 7]

x16 = [2 3 4 5 6] x17 = [1 3 4 5 6] x18 = [1 2 4 5 6] x19 = [1 2 3 5 6] x20 = [1 2 3 4 6] x21 = [1 2 3 4 5]

partition: 5 3 0 0 0 0 0 0 0

x1 = [2 4 6 7] x2 = [1 3 4 6 7] x3 = [2 3 5 6 7] x4 = [1 3 5 6 7] x5 = [1 2 4 5 6 7]

x6 = [2 4 5 7] x7 = [1 3 4 5 7] x8 = [2 3 5 7] x9 = [1 3 5 7] x10 = [1 2 4 5 7]

x11 = [2 3 4 6 7] x12 = [1 3 4 6 7] x13 = [1 2 4 6 7] x14 = [1 2 3 5 6 7] x15 = [2 4 5 6]

x16 = [1 3 4 5 6] x17 = [2 3 5 6] x18 = [1 3 5 6] x19 = [1 2 4 5 6] x20 = [2 3 4 6]

x21 = [1 3 4 6] x22 = [1 2 4 6] x23 = [1 2 3 5 6] x24 = [2 3 4 5 7] x25 = [1 3 4 5 7]

x26 = [1 2 4 5 7] x27 = [1 2 3 5 7] x28 = [1 2 3 4 6 7]

partition: 5 2 1 0 0 0 0 0 0

x1 = [3 5 6 7] x2 = [2 4 5 6 7] x3 = [1 4 5 6 7] x4 = [2 5 6 7] x5 = [1 3 5 6 7]

x6 = [3 4 6 7] x7 = [2 4 6 7] x8 = [1 4 6 7] x9 = [2 3 5 6 7] x10 = [1 3 5 6 7]

x11 = [1 2 5 6 7] x12 = [2 4 6 7] x13 = [1 3 4 6 7] x14 = [2 3 6 7] x15 = [1 3 6 7]

x16 = [1 2 4 6 7] x17 = [3 4 5 7] x18 = [2 4 5 7] x19 = [1 4 5 7] x20 = [2 3 5 7]

x21 = [1 3 5 7] x22 = [1 2 5 7] x23 = [2 3 4 6 7] x24 = [1 3 4 6 7] x25 = [1 2 4 6 7]

x26 = [1 2 3 6 7] x27 = [2 4 5 7] x28 = [1 3 4 5 7] x29 = [2 3 5 7] x30 = [1 3 5 7]

x31 = [1 2 4 5 7] x32 = [2 3 4 7] x33 = [1 3 4 7] x34 = [1 2 4 7] x35 = [1 2 3 5 7]

x36 = [3 4 5 6] x37 = [2 4 5 6] x38 = [1 4 5 6] x39 = [2 3 5 6] x40 = [1 3 5 6]

x41 = [1 2 5 6] x42 = [2 3 4 6] x43 = [1 3 4 6] x44 = [1 2 4 6] x45 = [1 2 3 6]

x46 = [2 3 4 5 7] x47 = [1 3 4 5 7] x48 = [1 2 4 5 7] x49 = [1 2 3 5 7] x50 = [1 2 3 4 7]

x51 = [2 4 5 6] x52 = [1 3 4 5 6] x53 = [2 3 5 6] x54 = [1 3 5 6] x55 = [1 2 4 5 6]

x56 = [2 3 4 6] x57 = [1 3 4 6] x58 = [1 2 4 6] x59 = [1 2 3 5 6] x60 = [2 3 4 5]

x61 = [1 3 4 5] x62 = [1 2 4 5] x63 = [1 2 3 5] x64 = [1 2 3 4 6]

partition: 5 1 1 1 0 0 0 0 0

x1 = [4 5 6 7] x2 = [3 5 6 7] x3 = [2 5 6 7] x4 = [1 5 6 7] x5 = [3 4 6 7]

x6 = [2 4 6 7] x7 = [1 4 6 7] x8 = [2 3 6 7] x9 = [1 3 6 7] x10 = [1 2 6 7]

x11 = [3 4 5 7] x12 = [2 4 5 7] x13 = [1 4 5 7] x14 = [2 3 5 7] x15 = [1 3 5 7]

x16 = [1 2 5 7] x17 = [2 3 4 7] x18 = [1 3 4 7] x19 = [1 2 4 7] x20 = [1 2 3 7]

x21 = [3 4 5 6] x22 = [2 4 5 6] x23 = [1 4 5 6] x24 = [2 3 5 6] x25 = [1 3 5 6]

x26 = [1 2 5 6] x27 = [2 3 4 6] x28 = [1 3 4 6] x29 = [1 2 4 6] x30 = [1 2 3 6]

x31 = [2 3 4 5] x32 = [1 3 4 5] x33 = [1 2 4 5] x34 = [1 2 3 5] x35 = [1 2 3 4]

partition: 4 4 0 0 0 0 0 0 0

x1 = [2 4 6] x2 = [1 3 4 6] x3 = [2 3 5 6] x4 = [1 3 5 6] x5 = [1 2 4 5 6]
x6 = [2 4 5 7] x7 = [1 3 4 5 7] x8 = [2 3 5 7] x9 = [1 3 5 7] x10 = [1 2 4 5 7]
x11 = [2 3 4 6 7] x12 = [1 3 4 6 7] x13 = [1 2 4 6 7] x14 = [1 2 3 5 6 7]
partition: 4 3 1 0 0 0 0 0 0
x1 = [3 5 7] x2 = [2 4 5 7] x3 = [1 4 5 7] x4 = [2 5 7] x5 = [1 3 5 7]
x6 = [3 4 6 7] x7 = [2 4 6 7] x8 = [1 4 6 7] x9 = [2 3 5 6 7] x10 = [1 3 5 6 7]
x11 = [1 2 5 6 7] x12 = [2 4 6 7] x13 = [1 3 4 6 7] x14 = [2 3 6 7] x15 = [1 3 6 7]
x16 = [1 2 4 6 7] x17 = [2 4 7] x18 = [1 3 4 7] x19 = [2 3 5 7] x20 = [1 3 5 7]
x21 = [1 2 4 5 7] x22 = [3 5 6] x23 = [2 4 5 6] x24 = [1 4 5 6] x25 = [2 5 6]
x26 = [1 3 5 6] x27 = [3 4 6] x28 = [2 4 6] x29 = [1 4 6] x30 = [2 3 5 6]
x31 = [1 3 5 6] x32 = [1 2 5 6] x33 = [2 4 6] x34 = [1 3 4 6] x35 = [2 3 6]
x36 = [1 3 6] x37 = [1 2 4 6] x38 = [3 4 5 7] x39 = [2 4 5 7] x40 = [1 4 5 7]
x41 = [2 3 5 7] x42 = [1 3 5 7] x43 = [1 2 5 7] x44 = [2 3 4 6 7] x45 = [1 3 4 6 7]
x46 = [1 2 4 6 7] x47 = [1 2 3 6 7] x48 = [2 4 5 7] x49 = [1 3 4 5 7] x50 = [2 3 5 7]
x51 = [1 3 5 7] x52 = [1 2 4 5 7] x53 = [2 3 4 7] x54 = [1 3 4 7] x55 = [1 2 4 7]
x56 = [1 2 3 5 7] x57 = [2 4 6] x58 = [1 3 4 6] x59 = [2 3 5 6] x60 = [1 3 5 6]
x61 = [1 2 4 5 6] x62 = [2 4 5] x63 = [1 3 4 5] x64 = [2 3 5] x65 = [1 3 5]
x66 = [1 2 4 5] x67 = [2 3 4 6] x68 = [1 3 4 6] x69 = [1 2 4 6] x70 = [1 2 3 5 6]
partition: 4 2 2 0 0 0 0 0 0
x1 = [3 6 7] x2 = [2 4 6 7] x3 = [1 4 6 7] x4 = [2 5 6 7] x5 = [1 3 5 6 7]
x6 = [3 5 7] x7 = [2 4 5 7] x8 = [1 4 5 7] x9 = [2 5 7] x10 = [1 3 5 7]
x11 = [3 4 7] x12 = [2 4 7] x13 = [1 4 7] x14 = [2 3 5 7] x15 = [1 3 5 7]
x16 = [1 2 5 7] x17 = [2 4 6 7] x18 = [1 3 4 6 7] x19 = [2 3 6 7] x20 = [1 3 6 7]
x21 = [1 2 4 6 7] x22 = [3 5 6] x23 = [2 4 5 6] x24 = [1 4 5 6] x25 = [2 5 6]
x26 = [1 3 5 6] x27 = [3 4 6] x28 = [2 4 6] x29 = [1 4 6] x30 = [2 3 5 6]
x31 = [1 3 5 6] x32 = [1 2 5 6] x33 = [2 4 6] x34 = [1 3 4 6] x35 = [2 3 6]
x36 = [1 3 6] x37 = [1 2 4 6] x38 = [3 4 5] x39 = [2 4 5] x40 = [1 4 5]
x41 = [2 3 5] x42 = [1 3 5] x43 = [1 2 5] x44 = [2 3 4 6] x45 = [1 3 4 6]
x46 = [1 2 4 6] x47 = [1 2 3 6] x48 = [2 4 5 7] x49 = [1 3 4 5 7] x50 = [2 3 5 7]
x51 = [1 3 5 7] x52 = [1 2 4 5 7] x53 = [2 3 4 7] x54 = [1 3 4 7] x55 = [1 2 4 7] x56 = [1 2 3 5 7]
partition: 4 2 1 1 0 0 0 0 0
x1 = [4 6 7] x2 = [3 5 6 7] x3 = [2 5 6 7] x4 = [1 5 6 7] x5 = [3 6 7]
x6 = [2 4 6 7] x7 = [1 4 6 7] x8 = [2 6 7] x9 = [1 3 6 7] x10 = [4 5 7]
x11 = [3 5 7] x12 = [2 5 7] x13 = [1 5 7] x14 = [3 4 6 7] x15 = [2 4 6 7]
x16 = [1 4 6 7] x17 = [2 3 6 7] x18 = [1 3 6 7] x19 = [1 2 6 7] x20 = [3 5 7]
x21 = [2 4 5 7] x22 = [1 4 5 7] x23 = [2 5 7] x24 = [1 3 5 7] x25 = [3 4 7]
x26 = [2 4 7] x27 = [1 4 7] x28 = [2 3 5 7] x29 = [1 3 5 7] x30 = [1 2 5 7]
x31 = [2 4 7] x32 = [1 3 4 7] x33 = [2 3 7] x34 = [1 3 7] x35 = [1 2 4 7]
x36 = [4 5 6] x37 = [3 5 6] x38 = [2 5 6] x39 = [1 5 6] x40 = [3 4 6]
x41 = [2 4 6] x42 = [1 4 6] x43 = [2 3 6] x44 = [1 3 6] x45 = [1 2 6]
x46 = [3 4 5 7] x47 = [2 4 5 7] x48 = [1 4 5 7] x49 = [2 3 5 7] x50 = [1 3 5 7]
x51 = [1 2 5 7] x52 = [2 3 4 7] x53 = [1 3 4 7] x54 = [1 2 4 7] x55 = [1 2 3 7]
x56 = [3 5 6] x57 = [2 4 5 6] x58 = [1 4 5 6] x59 = [2 5 6] x60 = [1 3 5 6]
x61 = [3 4 6] x62 = [2 4 6] x63 = [1 4 6] x64 = [2 3 5 6] x65 = [1 3 5 6]
x66 = [1 2 5 6] x67 = [2 4 6] x68 = [1 3 4 6] x69 = [2 3 6] x70 = [1 3 6]
x71 = [1 2 4 6] x72 = [3 4 5] x73 = [2 4 5] x74 = [1 4 5] x75 = [2 3 5]
x76 = [1 3 5] x77 = [1 2 5] x78 = [2 3 4 6] x79 = [1 3 4 6] x80 = [1 2 4 6]

x81 = [1 2 3 6] x82 = [2 4 5] x83 = [1 3 4 5] x84 = [2 3 5] x85 = [1 3 5]
x86 = [1 2 4 5] x87 = [2 3 4] x88 = [1 3 4] x89 = [1 2 4] x90 = [1 2 3 5]
partition: 4 1 1 1 1 0 0 0 0
x1 = [5 6 7] x2 = [4 6 7] x3 = [3 6 7] x4 = [2 6 7] x5 = [1 6 7]
x6 = [4 5 7] x7 = [3 5 7] x8 = [2 5 7] x9 = [1 5 7] x10 = [3 4 7]
x11 = [2 4 7] x12 = [1 4 7] x13 = [2 3 7] x14 = [1 3 7] x15 = [1 2 7]
x16 = [4 5 6] x17 = [3 5 6] x18 = [2 5 6] x19 = [1 5 6] x20 = [3 4 6]
x21 = [2 4 6] x22 = [1 4 6] x23 = [2 3 6] x24 = [1 3 6] x25 = [1 2 6]
x26 = [3 4 5] x27 = [2 4 5] x28 = [1 4 5] x29 = [2 3 5] x30 = [1 3 5]
x31 = [1 2 5] x32 = [2 3 4] x33 = [1 3 4] x34 = [1 2 4] x35 = [1 2 3]
partition: 3 3 2 0 0 0 0 0 0
x1 = [3 6] x2 = [2 4 6] x3 = [1 4 6] x4 = [2 5 6] x5 = [1 3 5 6]
x6 = [3 5 7] x7 = [2 4 5 7] x8 = [1 4 5 7] x9 = [2 5 7] x10 = [1 3 5 7]
x11 = [3 4 7] x12 = [2 4 7] x13 = [1 4 7] x14 = [2 3 5 7] x15 = [1 3 5 7]
x16 = [1 2 5 7] x17 = [2 4 6 7] x18 = [1 3 4 6 7] x19 = [2 3 6 7] x20 = [1 3 6 7]
x21 = [1 2 4 6 7] x22 = [3 5] x23 = [2 4 5] x24 = [1 4 5] x25 = [2 5]
x26 = [1 3 5] x27 = [3 4 6] x28 = [2 4 6] x29 = [1 4 6] x30 = [2 3 5 6]
x31 = [1 3 5 6] x32 = [1 2 5 6] x33 = [2 4 6] x34 = [1 3 4 6] x35 = [2 3 6] x36 = [1 3 6]
x37 = [1 2 4 6] x38 = [2 4 7] x39 = [1 3 4 7] x40 = [2 3 5 7] x41 = [1 3 5 7] x42 = [1 2 4 5 7]
partition: 3 3 1 1 0 0 0 0 0
x1 = [4 6] x2 = [3 5 6] x3 = [2 5 6] x4 = [1 5 6] x5 = [3 6]
x6 = [2 4 6] x7 = [1 4 6] x8 = [2 6] x9 = [1 3 6] x10 = [4 5 7]
x11 = [3 5 7] x12 = [2 5 7] x13 = [1 5 7] x14 = [3 4 6 7] x15 = [2 4 6 7]
x16 = [1 4 6 7] x17 = [2 3 6 7] x18 = [1 3 6 7] x19 = [1 2 6 7] x20 = [3 5 7]
x21 = [2 4 5 7] x22 = [1 4 5 7] x23 = [2 5 7] x24 = [1 3 5 7] x25 = [3 4 7]
x26 = [2 4 7] x27 = [1 4 7] x28 = [2 3 5 7] x29 = [1 3 5 7] x30 = [1 2 5 7]
x31 = [2 4 7] x32 = [1 3 4 7] x33 = [2 3 7] x34 = [1 3 7] x35 = [1 2 4 7]
x36 = [3 5] x37 = [2 4 5] x38 = [1 4 5] x39 = [2 5] x40 = [1 3 5]
x41 = [3 4 6] x42 = [2 4 6] x43 = [1 4 6] x44 = [2 3 5 6] x45 = [1 3 5 6]
x46 = [1 2 5 6] x47 = [2 4 6] x48 = [1 3 4 6] x49 = [2 3 6] x50 = [1 3 6]
x51 = [1 2 4 6] x52 = [2 4] x53 = [1 3 4] x54 = [2 3 5] x55 = [1 3 5] x56 = [1 2 4 5]
partition: 3 2 2 1 0 0 0 0 0
x1 = [4 7] x2 = [3 5 7] x3 = [2 5 7] x4 = [1 5 7] x5 = [3 6 7]
x6 = [2 4 6 7] x7 = [1 4 6 7] x8 = [2 6 7] x9 = [1 3 6 7] x10 = [3 7]
x11 = [2 4 7] x12 = [1 4 7] x13 = [2 5 7] x14 = [1 3 5 7] x15 = [4 6]
x16 = [3 5 6] x17 = [2 5 6] x18 = [1 5 6] x19 = [3 6] x20 = [2 4 6]
x21 = [1 4 6] x22 = [2 6] x23 = [1 3 6] x24 = [4 5] x25 = [3 5]
x26 = [2 5] x27 = [1 5] x28 = [3 4 6] x29 = [2 4 6] x30 = [1 4 6]
x31 = [2 3 6] x32 = [1 3 6] x33 = [1 2 6] x34 = [3 5 7] x35 = [2 4 5 7]
x36 = [1 4 5 7] x37 = [2 5 7] x38 = [1 3 5 7] x39 = [3 4 7] x40 = [2 4 7]
x41 = [1 4 7] x42 = [2 3 5 7] x43 = [1 3 5 7] x44 = [1 2 5 7] x45 = [2 4 7]
x46 = [1 3 4 7] x47 = [2 3 7] x48 = [1 3 7] x49 = [1 2 4 7] x50 = [3 6]
x51 = [2 4 6] x52 = [1 4 6] x53 = [2 5 6] x54 = [1 3 5 6] x55 = [3 5]
x56 = [2 4 5] x57 = [1 4 5] x58 = [2 5] x59 = [1 3 5] x60 = [3 4]
x61 = [2 4] x62 = [1 4] x63 = [2 3 5] x64 = [1 3 5] x65 = [1 2 5]
x66 = [2 4 6] x67 = [1 3 4 6] x68 = [2 3 6] x69 = [1 3 6] x70 = [1 2 4 6]
partition: 3 2 1 1 1 0 0 0 0

x1 = [5 7] x2 = [4 6 7] x3 = [3 6 7] x4 = [2 6 7] x5 = [1 6 7]
x6 = [4 7] x7 = [3 5 7] x8 = [2 5 7] x9 = [1 5 7] x10 = [3 7]
x11 = [2 4 7] x12 = [1 4 7] x13 = [2 7] x14 = [1 3 7] x15 = [5 6]
x16 = [4 6] x17 = [3 6] x18 = [2 6] x19 = [1 6] x20 = [4 5 7]
x21 = [3 5 7] x22 = [2 5 7] x23 = [1 5 7] x24 = [3 4 7] x25 = [2 4 7]
x26 = [1 4 7] x27 = [2 3 7] x28 = [1 3 7] x29 = [1 2 7] x30 = [4 6]
x31 = [3 5 6] x32 = [2 5 6] x33 = [1 5 6] x34 = [3 6] x35 = [2 4 6]
x36 = [1 4 6] x37 = [2 6] x38 = [1 3 6] x39 = [4 5] x40 = [3 5]
x41 = [2 5] x42 = [1 5] x43 = [3 4 6] x44 = [2 4 6] x45 = [1 4 6]
x46 = [2 3 6] x47 = [1 3 6] x48 = [1 2 6] x49 = [3 5] x50 = [2 4 5]
x51 = [1 4 5] x52 = [2 5] x53 = [1 3 5] x54 = [3 4] x55 = [2 4]
x56 = [1 4] x57 = [2 3 5] x58 = [1 3 5] x59 = [1 2 5] x60 = [2 4]
x61 = [1 3 4] x62 = [2 3] x63 = [1 3] x64 = [1 2 4]
partition: 3 1 1 1 1 1 0 0 0
x1 = [6 7] x2 = [5 7] x3 = [4 7] x4 = [3 7] x5 = [2 7]
x6 = [1 7] x7 = [5 6] x8 = [4 6] x9 = [3 6] x10 = [2 6]
x11 = [1 6] x12 = [4 5] x13 = [3 5] x14 = [2 5] x15 = [1 5]
x16 = [3 4] x17 = [2 4] x18 = [1 4] x19 = [2 3] x20 = [1 3] x21 = [1 2]
partition: 2 2 2 2 0 0 0 0 0
x1 = [4] x2 = [3 5] x3 = [2 5] x4 = [1 5] x5 = [3 6]
x6 = [2 4 6] x7 = [1 4 6] x8 = [2 6] x9 = [1 3 6] x10 = [3 7]
x11 = [2 4 7] x12 = [1 4 7] x13 = [2 5 7] x14 = [1 3 5 7]
partition: 2 2 2 1 1 0 0 0 0
x1 = [5] x2 = [4 6] x3 = [3 6] x4 = [2 6] x5 = [1 6]
x6 = [4 7] x7 = [3 5 7] x8 = [2 5 7] x9 = [1 5 7] x10 = [3 7]
x11 = [2 4 7] x12 = [1 4 7] x13 = [2 7] x14 = [1 3 7] x15 = [4]
x16 = [3 5] x17 = [2 5] x18 = [1 5] x19 = [3 6] x20 = [2 4 6]
x21 = [1 4 6] x22 = [2 6] x23 = [1 3 6] x24 = [3] x25 = [2 4]
x26 = [1 4] x27 = [2 5] x28 = [1 3 5]
partition: 2 2 1 1 1 1 0 0 0
x1 = [6] x2 = [5 7] x3 = [4 7] x4 = [3 7] x5 = [2 7]
x6 = [1 7] x7 = [5] x8 = [4 6] x9 = [3 6] x10 = [2 6]
x11 = [1 6] x12 = [4] x13 = [3 5] x14 = [2 5] x15 = [1 5]
x16 = [3] x17 = [2 4] x18 = [1 4] x19 = [2] x20 = [1 3]
partition: 2 1 1 1 1 1 1 0 0
x1 = [7] x2 = [6] x3 = [5] x4 = [4] x5 = [3] x6 = [2] x7 = [1]
partition: 1 1 1 1 1 1 1 1 0
Matrix size = 7, reduced Matrix size = 0
0 1 1 1 2 1 2 3 1 3 4 1 4 5 1 5 6 1
Matrix size = 20, reduced Matrix size = 0
0 1 1 0 2 1 0 4 2 1 3 1 2 3 1 2 5 1 3 4 1 3 6 1 3 8 2 4 7 1
5 6 1 5 9 1 6 7 1 6 10 1 7 8 1 7 11 1 7 13 2 8 12 1 9 10 1 9 14 1
10 11 1 10 15 1 11 12 1 11 16 1 12 13 1 12 17 1 12 19 2 13 18 1
14 15 1 15 16 1 16 17 1 17 18 1 18 19 1
Matrix size = 21, reduced Matrix size = 0
0 1 1 1 2 1 1 3 1 2 4 1 3 4 1 3 6 1 4 5 1 4 7 1 5 8 1
6 7 1 6 10 1 7 8 1 7 11 1 8 9 1 8 12 1 9 13 1 10 11 1 10 15 1

11 12 1 11 16 1 12 13 1 12 17 1 13 14 1 13 18 1 14 19 1 15 16 1
16 17 1 17 18 1 18 19 1 19 20 1
Matrix size = 28, reduced Matrix size = 1
0 1 1 0 2 1 0 4 2 0 5 1 0 10 2 0 12 3 1 3 1 1 6 1 1 11 2 2 3 1 2 7 1 2 13 4
3 4 1 3 8 1 4 9 1 5 6 1 5 7 1 5 9 2 5 14 1 6 8 1 6 15 1 7 8 1 7 10 1 7 16 1 7 23 2
8 9 1 8 11 1 8 13 2 8 17 1 8 24 2 8 26 3 9 12 1 9 18 1 9 25 2 10 11 1 10 19 1
11 12 1 11 20 1 11 27 4 12 13 1 12 21 1 13 22 1 14 15 1 14 16 1 14 18 2 15 17 1
16 17 1 16 19 1 17 18 1 17 20 1 17 22 2 18 21 1 19 20 1 19 23 1 20 21 1 20 24 1
21 22 1 21 25 1 21 27 2 22 26 1 23 24 1 24 25 1 25 26 1 26 27 1
Matrix size = 64, reduced Matrix size = 2
0 1 1 0 4 2 0 5 1 0 8 2 1 2 1 1 3 1 1 6 1 2 4 1 2 7 1 3 4 1 3 8 1 3 10 2 3 11 1
4 9 1 4 12 1 5 6 1 5 12 2 5 16 1 6 7 1 6 8 1 6 11 1 6 17 1 6 22 2
7 9 1 7 12 1 7 15 2 7 18 1 7 23 2 8 9 1 8 13 1 8 19 1 9 10 1 9 14 1 9 20 1
10 15 1 10 21 1 11 12 1 11 13 1 11 15 2 11 26 1 12 14 1 12 27 1 13 14 1 13 22 1 13 28 1
14 15 1 14 23 1 14 25 2 14 29 1 15 24 1 15 30 1 16 17 1 16 27 2 16 35 1
17 18 1 17 19 1 17 26 1 17 36 1 18 20 1 18 27 1 18 30 2 18 37 1
19 20 1 19 22 1 19 28 1 19 38 1 19 45 2 20 21 1 20 23 1 20 29 1 20 39 1 20 46 2
21 24 1 21 30 1 21 34 2 21 40 1 21 47 2 22 23 1 22 31 1 22 41 1
23 24 1 23 32 1 23 42 1 24 25 1 24 33 1 24 43 1 25 34 1 25 44 1
26 27 1 26 28 1 26 30 2 26 50 1 27 29 1 27 51 1 28 29 1 28 31 1 28 52 1
29 30 1 29 32 1 29 34 2 29 53 1 30 33 1 30 54 1 31 32 1 31 45 1 31 55 1
32 33 1 32 46 1 32 56 1 33 34 1 33 47 1 33 49 2 33 57 1 34 48 1 34 58 1
35 36 1 35 51 2 36 37 1 36 38 1 36 50 1 37 39 1 37 51 1 37 54 2 38 39 1 38 41 1 38 52 1
39 40 1 39 42 1 39 53 1 40 43 1 40 54 1 40 58 2 41 42 1 41 45 1 41 55 1
42 43 1 42 46 1 42 56 1 43 44 1 43 47 1 43 57 1 44 48 1 44 58 1 44 63 2
45 46 1 45 59 1 46 47 1 46 60 1 47 48 1 47 61 1 48 49 1 48 62 1 49 63 1
50 51 1 50 52 1 50 54 2 51 53 1 52 53 1 52 55 1 53 54 1 53 56 1 53 58 2
54 57 1 55 56 1 55 59 1 56 57 1 56 60 1 57 58 1 57 61 1 57 63 2 58 62 1
59 60 1 60 61 1 61 62 1 62 63 1
Matrix size = 35, reduced Matrix size = 1
0 1 1 1 2 1 1 4 1 2 3 1 2 5 1 3 6 1 4 5 1 4 10 1 5 6 1 5 7 1 5 11 1
6 8 1 6 12 1 7 8 1 7 13 1 8 9 1 8 14 1 9 15 1 10 11 1 10 20 1
11 12 1 11 13 1 11 21 1 12 14 1 12 22 1 13 14 1 13 16 1 13 23 1 14 15 1 14 17 1 14 24 1
15 18 1 15 25 1 16 17 1 16 26 1 17 18 1 17 27 1 18 19 1 18 28 1 19 29 1 20 21 1
21 22 1 21 23 1 22 24 1 23 24 1 23 26 1 24 25 1 24 27 1 25 28 1 26 27 1 26 30 1
27 28 1 27 31 1 28 29 1 28 32 1 29 33 1 30 31 1 31 32 1 32 33 1 33 34 1
Matrix size = 14, reduced Matrix size = 1
0 1 1 0 2 1 0 4 2 0 5 1 0 10 2 0 12 3 1 3 1 1 6 1 1 11 2 2 3 1 2 7 1 2 13 4
3 4 1 3 8 1 4 9 1 5 6 1 5 7 1 5 9 2 6 8 1 7 8 1 7 10 1 8 9 1 8 11 1 8 13 2
9 12 1 10 11 1 11 12 1 12 13 1
Matrix size = 70, reduced Matrix size = 4
0 1 1 0 4 2 0 5 1 0 8 2 0 21 1 0 37 2 0 40 3 1 2 1 1 3 1 1 6 1 1 22 1 1 38 2
2 4 1 2 7 1 2 20 3 2 23 1 2 39 2 3 4 1 3 8 1 3 10 2 3 11 1 3 18 2 3 20 3 3 24 1
3 40 2 3 42 3 3 47 2 4 9 1 4 12 1 4 19 2 4 25 1 4 41 2 4 48 2
5 6 1 5 12 2 5 26 1 5 43 4 6 7 1 6 8 1 6 11 1 6 27 1 7 9 1 7 12 1 7 15 2 7 28 1
8 9 1 8 13 1 8 29 1 9 10 1 9 14 1 9 30 1 10 15 1 10 31 1
11 12 1 11 13 1 11 15 2 11 16 1 11 32 1 11 43 3 11 45 4 12 14 1 12 17 1 12 33 1 12 44 3

13 14 1 13 18 1 13 34 1 13 46 5 14 15 1 14 19 1 14 35 1 15 20 1 15 36 1
16 17 1 16 18 1 16 20 2 16 47 1 16 52 2 16 54 3 16 56 1 17 19 1 17 48 1 17 53 2 17 57 1
18 19 1 18 49 1 18 55 4 18 58 1 19 20 1 19 50 1 19 59 1 20 51 1 20 60 1
21 22 1 21 25 2 21 26 1 21 29 2 22 23 1 22 24 1 22 27 1 23 25 1 23 28 1 23 60 3
24 25 1 24 29 1 24 31 2 24 32 1 24 58 2 24 60 3 25 30 1 25 33 1 25 59 2
26 27 1 26 33 2 26 37 1 27 28 1 27 29 1 27 32 1 27 38 1 27 43 2
28 30 1 28 33 1 28 36 2 28 39 1 28 44 2 29 30 1 29 34 1 29 40 1
30 31 1 30 35 1 30 41 1 31 36 1 31 42 1 31 69 3 32 33 1 32 34 1 32 36 2 32 47 1 32 56 1
33 35 1 33 48 1 33 57 1 34 35 1 34 43 1 34 49 1 34 58 1 34 66 2
35 36 1 35 44 1 35 46 2 35 50 1 35 59 1 35 67 2 35 69 3 36 45 1 36 51 1 36 60 1 36 68 2
37 38 1 37 48 2 38 39 1 38 40 1 38 47 1 39 41 1 39 48 1 39 51 2
40 41 1 40 43 1 40 49 1 41 42 1 41 44 1 41 50 1 42 45 1 42 51 1 42 55 2
43 44 1 43 52 1 44 45 1 44 53 1 45 46 1 45 54 1 46 55 1 47 48 1 47 49 1 47 51 2 47 61 1
48 50 1 48 62 1 49 50 1 49 52 1 49 63 1 50 51 1 50 53 1 50 55 2 50 64 1
51 54 1 51 65 1 52 53 1 52 66 1 53 54 1 53 67 1 54 55 1 54 68 1 55 69 1
56 57 1 56 58 1 56 60 2 56 61 1 56 66 2 56 68 3 57 59 1 57 62 1 57 67 2
58 59 1 58 63 1 58 69 4 59 60 1 59 64 1 60 65 1 61 62 1 61 63 1 61 65 2
62 64 1 63 64 1 63 66 1 64 65 1 64 67 1 64 69 2 65 68 1 66 67 1 67 68 1 68 69 1
Matrix size = 56, reduced Matrix size = 3
0 1 1 0 4 2 0 5 1 0 17 3 0 18 3 1 2 1 1 3 1 1 6 1 1 16 2 2 4 1 2 7 1 2 17 2
3 4 1 3 8 1 4 9 1 5 6 1 5 9 2 5 10 1 5 13 2 5 21 1 6 7 1 6 8 1 6 11 1 6 22 1
7 9 1 7 12 1 7 23 1 8 9 1 8 13 1 8 15 2 8 16 1 8 24 1 8 47 2
9 14 1 9 17 1 9 25 1 9 48 2 10 11 1 10 17 2 10 26 1 10 48 3
11 12 1 11 13 1 11 16 1 11 27 1 11 47 2 11 52 3
12 14 1 12 17 1 12 20 2 12 28 1 12 48 2 12 51 3 12 53 3
13 14 1 13 18 1 13 29 1 13 49 2 14 15 1 14 19 1 14 30 1 14 50 2
15 20 1 15 31 1 15 51 2 16 17 1 16 18 1 16 20 2 16 32 1
17 19 1 17 33 1 18 19 1 18 34 1 19 20 1 19 35 1 20 36 1
21 22 1 21 25 2 21 26 1 21 29 2 22 23 1 22 24 1 22 27 1 23 25 1 23 28 1
24 25 1 24 29 1 24 31 2 24 32 1 25 30 1 25 33 1 26 27 1 26 33 2 26 37 1
27 28 1 27 29 1 27 32 1 27 38 1 27 43 2 28 30 1 28 33 1 28 36 2 28 39 1 28 44 2
29 30 1 29 34 1 29 40 1 30 31 1 30 35 1 30 41 1 31 36 1 31 42 1
32 33 1 32 34 1 32 36 2 32 47 1 33 35 1 33 48 1 34 35 1 34 43 1 34 49 1
35 36 1 35 44 1 35 46 2 35 50 1 36 45 1 36 51 1 37 38 1 37 48 2
38 39 1 38 40 1 38 47 1 39 41 1 39 48 1 39 51 2 40 41 1 40 43 1 40 49 1
41 42 1 41 44 1 41 50 1 42 45 1 42 51 1 42 55 2 43 44 1 43 52 1
44 45 1 44 53 1 45 46 1 45 54 1 46 55 1 47 48 1 47 49 1 47 51 2
48 50 1 49 50 1 49 52 1 50 51 1 50 53 1 50 55 2 51 54 1 52 53 1 53 54 1 54 55 1
Matrix size = 90, reduced Matrix size = 5
0 1 1 0 5 2 0 9 1 0 13 2 1 2 1 1 4 1 1 10 1 2 3 1 2 5 1 2 11 1
3 6 1 3 12 1 4 5 1 4 8 2 4 13 1 4 16 2 4 19 1 5 6 1 5 7 1 5 14 1 5 20 1
6 8 1 6 15 1 6 21 1 7 8 1 7 16 1 7 18 2 7 22 1 8 17 1 8 23 1
9 10 1 9 20 2 9 35 1 10 11 1 10 13 1 10 19 1 10 36 1 10 45 2
11 12 1 11 14 1 11 20 1 11 27 2 11 37 1 11 46 2 12 15 1 12 21 1 12 28 2 12 38 1 12 47 2
13 14 1 13 24 1 13 39 1 14 15 1 14 16 1 14 25 1 14 40 1 15 17 1 15 26 1 15 41 1
16 17 1 16 27 1 16 42 1 17 18 1 17 28 1 17 43 1 18 29 1 18 44 1
19 20 1 19 23 2 19 24 1 19 27 2 19 55 1 20 21 1 20 22 1 20 25 1 20 56 1

21 23 1 21 26 1 21 57 1 22 23 1 22 27 1 22 29 2 22 30 1 22 58 1 23 28 1 23 31 1 23 59 1
24 25 1 24 31 2 24 45 1 24 60 1 25 26 1 25 27 1 25 30 1 25 46 1 25 51 2 25 61 1
26 28 1 26 31 1 26 34 2 26 47 1 26 52 2 26 62 1 27 28 1 27 32 1 27 48 1 27 63 1
28 29 1 28 33 1 28 49 1 28 64 1 29 34 1 29 50 1 29 65 1 30 31 1 30 32 1 30 34 2 30 66 1
31 33 1 31 67 1 32 33 1 32 51 1 32 68 1 33 34 1 33 52 1 33 54 2 33 69 1
34 53 1 34 70 1 35 36 1 35 56 2 36 37 1 36 39 1 36 55 1 37 38 1 37 40 1 37 56 1 37 63 2
38 41 1 38 57 1 38 64 2 39 40 1 39 45 1 39 60 1 40 41 1 40 42 1 40 46 1 40 61 1
41 43 1 41 47 1 41 62 1 42 43 1 42 48 1 42 63 1 42 77 2 43 44 1 43 49 1 43 64 1 43 78 2
44 50 1 44 65 1 44 79 2 45 46 1 45 71 1 46 47 1 46 48 1 46 72 1 47 49 1 47 73 1
48 49 1 48 51 1 48 74 1 49 50 1 49 52 1 49 75 1 50 53 1 50 76 1 51 52 1 51 77 1
52 53 1 52 78 1 53 54 1 53 79 1 54 80 1 55 56 1 55 59 2 55 60 1 55 63 2
56 57 1 56 58 1 56 61 1 57 59 1 57 62 1 58 59 1 58 63 1 58 65 2 58 66 1
59 64 1 59 67 1 60 61 1 60 67 2 60 71 1 61 62 1 61 63 1 61 66 1 61 72 1 61 77 2
62 64 1 62 67 1 62 70 2 62 73 1 62 78 2 63 64 1 63 68 1 63 74 1 64 65 1 64 69 1 64 75 1
65 70 1 65 76 1 66 67 1 66 68 1 66 70 2 66 81 1 67 69 1 67 82 1 68 69 1 68 77 1 68 83 1
69 70 1 69 78 1 69 80 2 69 84 1 70 79 1 70 85 1 71 72 1 71 82 2 72 73 1 72 74 1 72 81 1
73 75 1 73 82 1 73 85 2 74 75 1 74 77 1 74 83 1 75 76 1 75 78 1 75 84 1
76 79 1 76 85 1 76 89 2 77 78 1 77 86 1 78 79 1 78 87 1 79 80 1 79 88 1
80 89 1 81 82 1 81 83 1 81 85 2 82 84 1 83 84 1 83 86 1 84 85 1 84 87 1 84 89 2
85 88 1 86 87 1 87 88 1 88 89 1
Matrix size = 35, reduced Matrix size = 2
0 1 1 1 2 1 1 5 1 2 3 1 2 6 1 3 4 1 3 7 1 4 8 1 5 6 1 5 15 1
6 7 1 6 9 1 6 16 1 7 8 1 7 10 1 7 17 1 8 11 1 8 18 1 9 10 1 9 19 1
10 11 1 10 12 1 10 20 1 11 13 1 11 21 1 12 13 1 12 22 1 13 14 1 13 23 1
14 24 1 15 16 1 16 17 1 16 19 1 17 18 1 17 20 1 18 21 1 19 20 1 19 25 1
20 21 1 20 22 1 20 26 1 21 23 1 21 27 1 22 23 1 22 28 1 23 24 1 23 29 1
24 30 1 25 26 1 26 27 1 26 28 1 27 29 1 28 29 1 28 31 1 29 30 1 29 32 1
30 33 1 31 32 1 32 33 1 33 34 1
Matrix size = 42, reduced Matrix size = 3
0 1 1 0 4 2 0 5 1 0 17 3 0 18 3 0 26 2 0 29 3 0 35 5
1 2 1 1 3 1 1 6 1 1 16 2 1 27 2 1 36 4 2 4 1 2 7 1 2 17 2 2 28 2
3 4 1 3 8 1 3 29 2 3 31 3 4 9 1 4 30 2 5 6 1 5 9 2 5 10 1 5 13 2 5 21 1 5 40 4
6 7 1 6 8 1 6 11 1 6 22 1 6 41 5 7 9 1 7 12 1 7 23 1 8 9 1 8 13 1 8 15 2 8 16 1 8 24 1
9 14 1 9 17 1 9 25 1 10 11 1 10 17 2 10 26 1 10 38 3 11 12 1 11 13 1 11 16 1 11 27 1 11 37 2
12 14 1 12 17 1 12 20 2 12 28 1 12 38 2 12 41 3 13 14 1 13 18 1 13 29 1 13 39 2
14 15 1 14 19 1 14 30 1 14 40 2 15 20 1 15 31 1 15 41 2 16 17 1 16 18 1 16 20 2 16 32 1
17 19 1 17 33 1 18 19 1 18 34 1 19 20 1 19 35 1 20 36 1 21 22 1 21 25 2 21 26 1 21 29 2
22 23 1 22 24 1 22 27 1 23 25 1 23 28 1 23 41 3 24 25 1 24 29 1 24 31 2 24 32 1 24 39 2 24 41 3
25 30 1 25 33 1 25 40 2 26 27 1 26 33 2 27 28 1 27 29 1 27 32 1 28 30 1 28 33 1 28 36 2
29 30 1 29 34 1 30 31 1 30 35 1 31 36 1 32 33 1 32 34 1 32 36 2 32 37 1
33 35 1 33 38 1 34 35 1 34 39 1 35 36 1 35 40 1 36 41 1 37 38 1 37 39 1 37 41 2
38 40 1 39 40 1 40 41 1
Matrix size = 56, reduced Matrix size = 4
0 1 1 0 5 2 0 9 1 0 13 2 1 2 1 1 4 1 1 10 1 2 3 1 2 5 1 2 11 1 2 43 3
3 6 1 3 12 1 3 44 3 4 5 1 4 8 2 4 13 1 4 16 2 4 19 1 4 40 2 4 43 3
5 6 1 5 7 1 5 14 1 5 20 1 5 41 2 6 8 1 6 15 1 6 21 1 6 42 2
7 8 1 7 16 1 7 18 2 7 22 1 7 43 2 7 45 3 7 46 2 8 17 1 8 23 1 8 44 2 8 47 2

9 10 1 9 20 2 10 11 1 10 13 1 10 19 1 11 12 1 11 14 1 11 20 1 11 27 2
12 15 1 12 21 1 12 28 2 13 14 1 13 24 1 14 15 1 14 16 1 14 25 1 15 17 1 15 26 1
16 17 1 16 27 1 17 18 1 17 28 1 18 29 1 19 20 1 19 23 2 19 24 1 19 27 2 19 35 1
20 21 1 20 22 1 20 25 1 20 36 1 21 23 1 21 26 1 21 37 1 22 23 1 22 27 1 22 29 2 22 30 1 22 38 1
23 28 1 23 31 1 23 39 1 24 25 1 24 31 2 24 40 1 25 26 1 25 27 1 25 30 1 25 41 1
26 28 1 26 31 1 26 34 2 26 42 1 27 28 1 27 32 1 27 43 1 28 29 1 28 33 1 28 44 1
29 34 1 29 45 1 30 31 1 30 32 1 30 34 2 30 46 1 31 33 1 31 47 1 32 33 1 32 48 1
33 34 1 33 49 1 34 50 1 35 36 1 35 39 2 35 40 1 35 43 2 36 37 1 36 38 1 36 41 1
37 39 1 37 42 1 37 55 3 38 39 1 38 43 1 38 45 2 38 46 1 38 53 2 38 55 3
39 44 1 39 47 1 39 54 2 40 41 1 40 47 2 41 42 1 41 43 1 41 46 1 42 44 1 42 47 1 42 50 2
43 44 1 43 48 1 44 45 1 44 49 1 45 50 1 46 47 1 46 48 1 46 50 2 46 51 1
47 49 1 47 52 1 48 49 1 48 53 1 49 50 1 49 54 1 50 55 1 51 52 1 51 53 1 51 55 2
52 54 1 53 54 1 54 55 1
Matrix size = 70, reduced Matrix size = 5
0 1 1 0 5 2 0 11 4 0 14 1 0 34 3 0 38 3 1 2 1 1 4 1 1 13 3 1 15 1 1 33 2
2 3 1 2 5 1 2 12 2 2 16 1 2 34 2 3 6 1 3 13 2 3 17 1 3 35 2
4 5 1 4 8 2 4 9 1 4 18 1 5 6 1 5 7 1 5 10 1 5 19 1 6 8 1 6 11 1 6 20 1
7 8 1 7 12 1 7 21 1 8 13 1 8 22 1 9 10 1 9 13 2 9 33 1 9 45 3 9 46 3 9 49 1
10 11 1 10 12 1 10 34 1 10 44 2 10 50 1 11 13 1 11 35 1 11 45 2 11 51 1
12 13 1 12 36 1 12 52 1 13 37 1 13 53 1 14 15 1 14 19 2 14 23 1 14 27 2 14 51 4
15 16 1 15 18 1 15 24 1 15 53 3 16 17 1 16 19 1 16 25 1 16 52 2
17 20 1 17 26 1 17 53 2 18 19 1 18 22 2 18 27 1 18 30 2 18 33 1 18 49 1
19 20 1 19 21 1 19 28 1 19 34 1 19 50 1 20 22 1 20 29 1 20 35 1 20 51 1
21 22 1 21 30 1 21 32 2 21 36 1 21 52 1 21 65 2 22 31 1 22 37 1 22 53 1 22 66 2
23 24 1 23 34 2 23 56 5 24 25 1 24 27 1 24 33 1 24 58 4
25 26 1 25 28 1 25 34 1 25 41 2 25 57 3 26 29 1 26 35 1 26 42 2 26 58 3 26 64 4
27 28 1 27 38 1 27 66 3 28 29 1 28 30 1 28 39 1 28 65 2 29 31 1 29 40 1 29 66 2 29 69 3
30 31 1 30 41 1 30 67 2 31 32 1 31 42 1 31 68 2 32 43 1 32 69 2
33 34 1 33 37 2 33 38 1 33 41 2 33 54 1 34 35 1 34 36 1 34 39 1 34 55 1
35 37 1 35 40 1 35 56 1 36 37 1 36 41 1 36 43 2 36 44 1 36 57 1
37 42 1 37 45 1 37 58 1 38 39 1 38 45 2 38 59 1 39 40 1 39 41 1 39 44 1 39 60 1
40 42 1 40 45 1 40 48 2 40 61 1 41 42 1 41 46 1 41 62 1 42 43 1 42 47 1 42 63 1
43 48 1 43 64 1 44 45 1 44 46 1 44 48 2 44 65 1 45 47 1 45 66 1 46 47 1 46 67 1
47 48 1 47 68 1 48 69 1 49 50 1 49 53 2 49 54 1 49 66 3 49 67 3
50 51 1 50 52 1 50 55 1 50 65 2 51 53 1 51 56 1 51 66 2 52 53 1 52 57 1
53 58 1 54 55 1 54 58 2 54 59 1 54 62 2 55 56 1 55 57 1 55 60 1 56 58 1 56 61 1
57 58 1 57 62 1 57 64 2 57 65 1 58 63 1 58 66 1 59 60 1 59 66 2 60 61 1 60 62 1 60 65 1
61 63 1 61 66 1 61 69 2 62 63 1 62 67 1 63 64 1 63 68 1 64 69 1
65 66 1 65 67 1 65 69 2 66 68 1 67 68 1 68 69 1
Matrix size = 64, reduced Matrix size = 4
0 1 1 0 6 2 0 14 1 0 19 2 1 2 1 1 5 1 1 15 1 2 3 1 2 6 1 2 16 1
3 4 1 3 7 1 3 17 1 4 8 1 4 18 1 5 6 1 5 10 2 5 19 1 5 23 2 5 29 1
6 7 1 6 9 1 6 20 1 6 30 1 7 8 1 7 10 1 7 21 1 7 31 1 8 11 1 8 22 1 8 32 1
9 10 1 9 13 2 9 23 1 9 26 2 9 33 1 10 11 1 10 12 1 10 24 1 10 34 1
11 13 1 11 25 1 11 35 1 12 13 1 12 26 1 12 28 2 12 36 1 13 27 1 13 37 1
14 15 1 14 30 2 15 16 1 15 19 1 15 29 1 16 17 1 16 20 1 16 30 1 16 42 2
17 18 1 17 21 1 17 31 1 17 43 2 18 22 1 18 32 1 18 44 2

19 20 1 19 38 1 20 21 1 20 23 1 20 39 1 21 22 1 21 24 1 21 40 1 22 25 1 22 41 1
23 24 1 23 42 1 24 25 1 24 26 1 24 43 1 25 27 1 25 44 1 26 27 1 26 45 1
27 28 1 27 46 1 28 47 1 29 30 1 29 34 2 29 38 1 29 42 2 30 31 1 30 33 1 30 39 1
31 32 1 31 34 1 31 40 1 32 35 1 32 41 1 33 34 1 33 37 2 33 42 1 33 45 2 33 48 1
34 35 1 34 36 1 34 43 1 34 49 1 35 37 1 35 44 1 35 50 1 36 37 1 36 45 1 36 47 2 36 51 1
37 46 1 37 52 1 38 39 1 38 49 2 39 40 1 39 42 1 39 48 1 40 41 1 40 43 1 40 49 1 40 56 2
41 44 1 41 50 1 41 57 2 42 43 1 42 53 1 43 44 1 43 45 1 43 54 1 44 46 1 44 55 1
45 46 1 45 56 1 46 47 1 46 57 1 47 58 1 48 49 1 48 52 2 48 53 1 48 56 2
49 50 1 49 51 1 49 54 1 50 52 1 50 55 1 51 52 1 51 56 1 51 58 2 51 59 1
52 57 1 52 60 1 53 54 1 53 60 2 54 55 1 54 56 1 54 59 1 55 57 1 55 60 1 55 63 2
56 57 1 56 61 1 57 58 1 57 62 1 58 63 1 59 60 1 59 61 1 59 63 2 60 62 1 61 62 1 62 63 1
Matrix size = 21, reduced Matrix size = 1
0 1 1 1 2 1 1 6 1 2 3 1 2 7 1 3 4 1 3 8 1 4 5 1 4 9 1 5 10 1 6 7 1
7 8 1 7 11 1 8 9 1 8 12 1 9 10 1 9 13 1 10 14 1 11 12 1 12 13 1 12 15 1
13 14 1 13 16 1 14 17 1 15 16 1 16 17 1 16 18 1 17 19 1 18 19 1 19 20 1
Matrix size = 14, reduced Matrix size = 1
0 1 1 0 5 2 0 11 4 1 2 1 1 4 1 1 13 3 2 3 1 2 5 1 2 12 2 3 6 1 3 13 2
4 5 1 4 8 2 4 9 1 5 6 1 5 7 1 5 10 1 6 8 1 6 11 1 7 8 1 7 12 1 8 13 1
9 10 1 9 13 2 10 11 1 10 12 1 11 13 1 12 13 1
Matrix size = 28, reduced Matrix size = 2
0 1 1 0 6 2 0 16 4 1 2 1 1 5 1 1 19 3 2 3 1 2 6 1 2 18 2 3 4 1 3 7 1 3 19 2
4 8 1 4 20 2 5 6 1 5 10 2 5 14 1 6 7 1 6 9 1 6 15 1 7 8 1 7 10 1 7 16 1
8 11 1 8 17 1 9 10 1 9 13 2 9 18 1 10 11 1 10 12 1 10 19 1 11 13 1 11 20 1
12 13 1 12 21 1 13 22 1 14 15 1 14 19 2 14 25 4 15 16 1 15 18 1 15 27 3
16 17 1 16 19 1 16 26 2 17 20 1 17 27 2 18 19 1 18 22 2 18 23 1 19 20 1 19 21 1 19 24 1
20 22 1 20 25 1 21 22 1 21 26 1 22 27 1 23 24 1 23 27 2 24 25 1 24 26 1 25 27 1 26 27 1
Matrix size = 20, reduced Matrix size = 1
0 1 1 0 7 2 1 2 1 1 6 1 2 3 1 2 7 1 3 4 1 3 8 1 4 5 1 4 9 1 5 10 1
6 7 1 6 12 2 7 8 1 7 11 1 8 9 1 8 12 1 9 10 1 9 13 1 10 14 1 11 12 1 11 16 2
12 13 1 12 15 1 13 14 1 13 16 1 14 17 1 15 16 1 15 19 2 16 17 1 16 18 1 17 19 1 18 19 1
Matrix size = 7, reduced Matrix size = 0
0 1 1 1 2 1 2 3 1 3 4 1 4 5 1 5 6 1

```
#include      <stdio.h>
#define       max_rep_size          200
#define       max_matrix_size       25000
#define       size_tableau          200
#define       max_index             100
#define       red_matrix_size       500
typedef struct wptr {
   short      index;
   short      a2;
   struct wptr   *next;
}  WPTR;
char          *tableau[size_tableau];
char          *local_alpha, *alpha[max_rep_size];
unsigned int  *local_I_set, *I_set[max_rep_size];
WPTR          *incident_matrix[max_matrix_size];
int           g_l, g_bl, g_num_of_rep, g_pos, g_I_pos, g_num_vertex;
int           g_matrix_size[size_tableau];
int           real_max_matrix_size;
int           g_depth[size_tableau], max_depth, dim;
int           g_red_entry[red_matrix_size];
int           g_WI = 0;
FILE          *g_fp_Y;

int I_search_number(j, i)
int      j, i;
{
   register int   r, len;
   len = g_depth[j+1];
   for (r = 0; r < len; r++) if (i == (int)(tableau[r][j]-'@')) return(r);
   return(-1);
}
int get_number(i, k)
int      i, k;
{
   register int   p, q;
   for (p = 0; p <= g_num_of_rep; p++)
      for (q = k; q < g_depth[p+1]; q++) if (i == (int)(tableau[q][p]-'@')) return(1);
   return(0);
}
void I_set_bit(i)
int      i;
{
   switch (i) {
   case 1:
      local_I_set[g_I_pos] |= 0x0001; break;
   case 2:
      local_I_set[g_I_pos] |= 0x0002; break;
```

```
    case 3:
      local_I_set[g_I_pos] |= 0x0004; break;
    case 4:
      local_I_set[g_I_pos] |= 0x0008; break;
    case 5:
      local_I_set[g_I_pos] |= 0x0010; break;
    case 6:
      local_I_set[g_I_pos] |= 0x0020; break;
    case 7:
      local_I_set[g_I_pos] |= 0x0040; break;
    case 8:
      local_I_set[g_I_pos] |= 0x0080; break;
    case 9:
      local_I_set[g_I_pos] |= 0x0100; break;
    case 10:
      local_I_set[g_I_pos] |= 0x0200; break;
    case 11:
      local_I_set[g_I_pos] |= 0x0400; break;
    case 12:
      local_I_set[g_I_pos] |= 0x0800; break;
    case 13:
      local_I_set[g_I_pos] |= 0x1000; break;
    case 14:
      local_I_set[g_I_pos] |= 0x2000; break;
    case 15:
      local_I_set[g_I_pos] |= 0x4000; break;
    }
}
int get_class_number(i, p)
int     i, p;
{
    switch (i) {
    case 1:
      if (local_I_set[p] & 0x0001) return(1); break;
    case 2:
      if (local_I_set[p] & 0x0002) return(2); break;
    case 3:
      if (local_I_set[p] & 0x0004) return(3); break;
    case 4:
      if (local_I_set[p] & 0x0008) return(4); break;
    case 5:
      if (local_I_set[p] & 0x0010) return(5); break;
    case 6:
      if (local_I_set[p] & 0x0020) return(6); break;
    case 7:
      if (local_I_set[p] & 0x0040) return(7); break;
    case 8:
```

```
        if (local_I_set[p] & 0x0080) return(8); break;
    case 9:
        if (local_I_set[p] & 0x0100) return(9); break;
    case 10:
        if (local_I_set[p] & 0x0200) return(10); break;
    case 11:
        if (local_I_set[p] & 0x0400) return(11); break;
    case 12:
        if (local_I_set[p] & 0x0800) return(12); break;
    case 13:
        if (local_I_set[p] & 0x1000) return(13); break;
    case 14:
        if (local_I_set[p] & 0x2000) return(14); break;
    case 15:
        if (local_I_set[p] & 0x4000) return(15); break;
    }
    return(0);
}
void I_complement(I_class)
char    *I_class;
{
    register int    i, j;
    for (i = 1; i < g_bl; i++) {
        for (j = 0; I_class[j] != '@'; j++)
            if (i == (int)(I_class[j]-'@')) I_set_bit(i);
    }
}
void I_tableau()
{
    register int    i, j, k, pos;
    char            I_class[max_rep_size];
    pos = 0;
    if (g_num_of_rep == 0) {
        for (i = 1; i < g_bl; i++) I_class[pos++] = (char)(i + 64);
        I_class[pos] = '@';
    }
    else {
        for (i = 1; i < g_bl; i++) {
            for (j = 0; j <= g_num_of_rep; j++) {
                if ((k = I_search_number(j, i)) >= 0) {
                    if (get_number(i+1, k+1)) {
                        I_class[pos++] = (char)(i + 64);
                        break;
                    }
                }
            }
        }
```

```
    I_class[pos] = '@';
  }
  local_I_set[g_I_pos] = 0; /*   bit clear   */
  I_complement(I_class); g_I_pos++;
}
void alpha_tableau()
{
  register int   i, j;
  if (g_num_of_rep == 0) {
    for (i = 0; i < g_depth[1]; i++) local_alpha[g_pos++] = tableau[i][0];
    g_matrix_size[0]++;
  } else {
    for (i = g_l; i > 0; i--)
      for (j = 0; j < g_depth[i]; j++) local_alpha[g_pos++] = tableau[j][i-1];
    g_matrix_size[g_num_of_rep]++;
  }
  if (g_pos % g_bl != 0) {
    printf("mismatch in [alpha_tableau].¥n");
    exit(0);
  }
}
void young_tableau(bl, depth)
int          bl, depth[];
{
  int          i, j;
  char       temp;
  if (bl == 1) {
    tableau[0][0] = 'A'; alpha_tableau(); I_tableau();
    if (g_WI) {
      printf("x%01d = [", ++g_num_vertex);
      for (i = 1; i < g_bl; i++) {
        j = get_class_number(i, g_I_pos-1);
        if (j) printf("%01d ", j);
      }
      if (g_num_vertex % 5) printf("] ");
      else printf("]¥n");
    }
  } else {
    for (i = 0; depth[i] > 0; i++) {
      if (depth[i] > depth[i+1]) {
        temp = tableau[depth[i]-1][i];
        tableau[depth[i]-1][i] = (char)(bl + 64);
        depth[i]--;
        young_tableau(bl-1, depth);
        depth[i]++;
        tableau[depth[i]-1][i] = temp;
      }
```

```
int check_identity(w1, w2)
char   w1[], w2[];
{
  register int   i;
  i = 0;
  while (i < g_bl) {
    if (w1[i] != w2[i]) return(0);
    else i++;
  }
  return(1);
}
int phase1_check(w1, w2)
char   w1[], w2[];
{
  register int    i, j;
  for (i = 0; i < g_bl-2; i++) {       /* 93, 4/30    */
    for (j = i+2; j < g_bl; j++) {
      if (pair_between(w1, i, j)) {
        ch_exchange(&w1[i], &w1[j]);
        if (check_identity(w1, w2)) return(1);
        ch_exchange(&w1[i], &w1[j]);
      }
    }
  }
  return(0);
}
int search_number_position(i, w)
int    i;
char   w[];
{
  register int   j;
  for (j = 0; j < g_bl; j++)
    if (i == (int)(w[j]-'@')) return(j);
  printf("An error has happened in [search_number_position].¥n");
  exit(0);
}
int search_word_position(cp, w)
int    cp;
char   w[];
{
  register int    i, k, p;
  for (i = 0; i < g_matrix_size[cp]; i++) {
    for (p = 0, k = g_bl*i; k < g_bl*(i+1); k++, p++) {
      if (w[p] != alpha[cp][k]) {
        p = 0;
        break;
      }
```

```
    }
    if (p > 0) return(i);
  }
  printf("An error happened in [search_word_position].¥n");
  exit(0);
}
void set_W_graph_entry(i, j, mp)
int     i, j, mp;
{
  WPTR   *temp, *inf;
  if (j < i) exchange(&i, &j);
  temp = incident_matrix[i];
  if (temp == NULL) {
    incident_matrix[i] = (WPTR *)malloc(sizeof(WPTR));
    incident_matrix[i]->index = mp;
    incident_matrix[i]->a2 = j;
    incident_matrix[i]->next = NULL;
  }
  else {
    if (j < temp->a2) {
      temp = incident_matrix[i]->next;
      incident_matrix[i] = (WPTR *)malloc(sizeof(WPTR));
      incident_matrix[i]->index = mp;
      incident_matrix[i]->a2 = j;
      incident_matrix[i]->next = temp;
      return;
    }
    while (temp != NULL) {
      if (temp->a2 == j) {
        temp->index = mp;
        return;
      }
      else if (temp->a2 < j) {
          if (temp->next == NULL) {
            temp->next = (WPTR *)malloc(sizeof(WPTR));
            temp->next->index = mp;
            temp->next->a2 = j;
            temp->next->next = NULL;
            return;
          }
        else if (j < temp->next->a2) {
            inf = temp->next;
            temp->next = (WPTR *)malloc(sizeof(WPTR));
            temp->next->index = mp;
            temp->next->a2 = j;
            temp->next->next = inf;
            return;
```

```
            }
          }
          temp = temp->next;
        }
    }
}
void phase2_check(cp, mp, w1, w2)
int     cp, mp;
char    w1[], w2[];
{
    register int    i;
    int             p1, p2, p3, q1, q2, q3, s1, s2;

    for (i = 1; i <= g_bl-2; i++) {
        p1 = search_number_position(i, w1);
        p2 = search_number_position(i+1, w1);
        p3 = search_number_position(i+2, w1);
        if ((p1 < p2 && p2 < p3) || (p3 < p2 && p2 < p1)) ;
        else {
            q1 = search_number_position(i, w2);
            q2 = search_number_position(i+1, w2);
            q3 = search_number_position(i+2, w2);
            if ((q1 < q2 && q2 < q3) || (q3 < q2 && q2 < q1)) ;
            else {
                if ((p1 < p3 && p3 < p2) || (p2 < p3 && p3 < p1))
                    ch_exchange(&w1[p1], &w1[p2]);
                else if ((p2 < p1 && p1 < p3) || (p3 < p1 && p1 < p2))
                    ch_exchange(&w1[p2], &w1[p3]);

                if ((q1 < q3 && q3 < q2) || (q2 < q3 && q3 < q1))
                    ch_exchange(&w2[q1], &w2[q2]);
                else if ((q2 < q1 && q1 < q3) || (q3 < q1 && q1 < q2))
                    ch_exchange(&w2[q2], &w2[q3]);

                s1 = search_word_position(cp, w1);
                s2 = search_word_position(cp, w2);

                if (!check_W_graph_entry(s1, s2, 0))
                    set_W_graph_entry(s1, s2, mp+1);

                if ((p1 < p3 && p3 < p2) || (p2 < p3 && p3 < p1))
                    ch_exchange(&w1[p1], &w1[p2]);
                else if ((p2 < p1 && p1 < p3) || (p3 < p1 && p1 < p2))
                    ch_exchange(&w1[p2], &w1[p3]);

                if ((q1 < q3 && q3 < q2) || (q2 < q3 && q3 < q1))
                    ch_exchange(&w2[q1], &w2[q2]);
```

```
            else if ((q2 < q1 && q1 < q3) || (q3 < q1 && q1 < q2))
              ch_exchange(&w2[q2], &w2[q3]);
          }
        }
      }
    }
}
void make_W_graph1(cp, i, j)
int       cp, i, j;
{
  register int    k, p;
  char            word1[size_tableau], word2[size_tableau];
  for (p = 0, k = g_bl*i; k < g_bl*(i+1); k++, p++) word1[p] = alpha[cp][k];
  for (p = 0, k = g_bl*j; k < g_bl*(j+1); k++, p++) word2[p] = alpha[cp][k];
  if (phase1_check(word1, word2)) set_W_graph_entry(i, j, 1);
}
void make_W_graph2(cp, mp, i, j)
int       cp, mp, i, j;
{
  register int    k, p;
  char            word1[size_tableau], word2[size_tableau];
  for (p = 0, k = g_bl*i; k < g_bl*(i+1); k++, p++) word1[p] = alpha[cp][k];
  for (p = 0, k = g_bl*j; k < g_bl*(j+1); k++, p++) word2[p] = alpha[cp][k];
  phase2_check(cp, mp, word1, word2);
}
int inclusion(k, j, m)
int       k, j, m;
{
  int       res;
  switch (j) {
  case 1:
    res = I_set[k][m] & 0x0001; break;
  case 2:
    res = I_set[k][m] & 0x0002; break;
  case 3:
    res = I_set[k][m] & 0x0004; break;
  case 4:
    res = I_set[k][m] & 0x0008; break;
  case 5:
    res = I_set[k][m] & 0x0010; break;
  case 6:
    res = I_set[k][m] & 0x0020; break;
  case 7:
    res = I_set[k][m] & 0x0040; break;
  case 8:
    res = I_set[k][m] & 0x0080; break;
  case 9:
    res = I_set[k][m] & 0x0100; break;
```

```
    case 10:
      res = I_set[k][m] & 0x0200; break;
    case 11:
      res = I_set[k][m] & 0x0400; break;
    case 12:
      res = I_set[k][m] & 0x0800; break;
    case 13:
      res = I_set[k][m] & 0x1000; break;
    case 14:
      res = I_set[k][m] & 0x2000; break;
    case 15:
      res = I_set[k][m] & 0x4000; break;
    }
    return(res);
}
int check_W_graph_entry(i, j, k)
int      i, j, k;
{
    WPTR   *temp;
    if (j < i) exchange(&i, &j);
    temp = incident_matrix[i];
    if (temp == NULL) return(0);
    else {
      if (j < temp->a2) return(0);
      while (temp != NULL) {
        if (temp->a2 == j) {
          if (k == 0) return(1);
          else if (temp->index == k) return(k);
          else return(0);
        }
        else if (temp->a2 < j) {
          if (temp->next == NULL) return(0);
          else if (j < temp->next->a2) return(0);
        }
        temp = temp->next;
      }
    }
    return(0);
}
void
make_representation(k, s)
int     k, s;
{
    int      j, m, n;
    char   buf[50];
    FILE    *fp_w;
    for (j = 0; j < g_bl-1; j++) {
```

```
      sprintf(buf, "R%02d%02d%02d", g_bl, k, j+1);
      if ((fp_w = fopen(buf, "w")) == NULL) {
        printf("File %s can not open.¥n", buf);
        exit(0);
      }
      for (n = 0; n < s; n++) {
        for (m = 0; m < n; m++) {
          if (inclusion(k, j+1, m) && !inclusion(k, j+1, n) && check_W_graph_entry(m, n, 0))
            fprintf(fp_w, "%1d %1d 1 ", m, n);
        }
        if (inclusion(k, j+1, n)) fprintf(fp_w, "%1d %1d -1 ", n, n);
        else fprintf(fp_w, "%1d %1d 2 ", n, n);
        for (m = n+1; m < s; m++) {
          if (inclusion(k, j+1, m) && !inclusion(k, j+1, n) && check_W_graph_entry(m, n, 0))
            fprintf(fp_w, "%1d %1d 1 ", m, n);
        }
        fprintf(fp_w, "¥n");
      }
      fprintf(fp_w, "%d %d¥n", 9999, 9999);
      fclose(fp_w);
    }
  }
int get_partition(n)
int     n;
{
  int   i;
  int   depth[size_tableau];
  if (n > 0) {
    for (i = g_depth[g_l]; i > 0; i--) {
      g_l++;
      g_depth[g_l] = i;
      get_partition(n-i);
      g_l--;
    }
  } else if (n == 0) {
    printf("¥npartition: ");
    for (i = 0; i <= g_bl; i++) {
      if (i < g_l) depth[i] =  g_depth[i+1];
      else depth[i] = 0;
      printf(" %d", depth[i]);
    }
    if (depth[max_depth] != 0) printf("¥n");
    else {
      for (i = 0; i < max_depth; i++) fprintf(g_fp_Y, " %d", depth[i]);
      fprintf(g_fp_Y, "¥n");
      printf("¥n");
      g_matrix_size[g_num_of_rep] = g_pos = g_I_pos = g_num_vertex = 0;
```

```
        young_tableau(g_bl, depth);
        alpha[g_num_of_rep] = (char *)malloc((long)sizeof(char)*g_matrix_size[g_num_of_rep]*g_bl);
        if (alpha[g_num_of_rep] == NULL) {
          printf("Memory exhausted in [alpha[%d]].¥n", g_num_of_rep);
          exit(0);
        }
        I_set[g_num_of_rep] = (unsigned int *)malloc((long)sizeof(int)*g_matrix_size[g_num_of_rep]);
        if (I_set[g_num_of_rep] == NULL) {
          printf("Memory exhausted in [I_set[%d]].¥n", g_num_of_rep);
          exit(0);
        }
        for (i = 0; i < g_matrix_size[g_num_of_rep]*g_bl; i++)
          alpha[g_num_of_rep][i] = local_alpha[i];
        for (i = 0; i < g_matrix_size[g_num_of_rep]; i++)
          I_set[g_num_of_rep][i] = local_I_set[i];
        if (real_max_matrix_size < g_matrix_size[g_num_of_rep])
          real_max_matrix_size = g_matrix_size[g_num_of_rep];
        g_num_of_rep++;
      }
    }
}
int partition(n)
int    n;
{
  g_bl = g_depth[0] = n;
  g_l = g_num_of_rep = 0;
  local_alpha = (char *)malloc((long)sizeof(char)*max_rep_size*max_matrix_size);
  local_I_set = (unsigned int *)malloc((long)sizeof(int)*max_rep_size*max_matrix_size);
  get_partition(n); free(local_alpha); free(local_I_set);
}
void write_W_matrix(k)
int            k;
{
   int       m;
  WPTR   *temp;
  printf("Matrix size = %d, reduced Matrix size = %d¥n", g_matrix_size[k], dim);
  if (g_WI < 2) return;
  for (m = 0; m < g_matrix_size[k]; m++) {
    temp = incident_matrix[m];
    while (temp != NULL) {
    /*   printf("%2d %2d %2d ", m, temp->a2, temp->index);   */
      printf("%2d %2d ", m, temp->a2);
      temp = temp->next;
    }
     printf("¥n");
  }
  printf("¥n");
```

```
}
void match_braid_index(k, i)
int      k, i;
{
  switch (g_bl) {
    case 6:
    if ((I_set[k][i]&0x0012) == 0x0012 && !(I_set[k][i]&0x009)) g_red_entry[dim++] = i;
    break;
    case 7:
    if ((I_set[k][i]&0x0012) == 0x0012 && !(I_set[k][i]&0x009)) g_red_entry[dim++] = i;
    break;
    case 8:
    if ((I_set[k][i]&0x0012) == 0x0012 && !(I_set[k][i]&0x0049)) g_red_entry[dim++] = i;
    break;
    case 9:
    if ((I_set[k][i]&0x0092) == 0x0092 && !(I_set[k][i]&0x0049)) g_red_entry[dim++] = i;
    break;
    case 10:
    if ((I_set[k][i]&0x0092) == 0x0092 && !(I_set[k][i]&0x0049)) g_red_entry[dim++] = i;
    break;
    case 11:
    if ((I_set[k][i]&0x0092) == 0x0092 && !(I_set[k][i]&0x0249)) g_red_entry[dim++] = i;
    break;
    case 12:
    if ((I_set[k][i]&0x0492) == 0x0492 && !(I_set[k][i]&0x0249)) g_red_entry[dim++] = i;
    break;
  }
}
void reduction_space(k, s)
int            k, s;
{
  int    i;
  for (i = 0; i < s; i++) match_braid_index(k, i);
}
static void release_memory()
{
  int            i;
  for (i = 0; i < g_num_of_rep; i++) free(alpha[i]);
  for (i = 0; i < g_num_of_rep; i++) free(I_set[i]);
}
main()
{
  int      i, j, k, m, bl, p0, p1;
  char   buf[50];
  FILE   *fp_w;
  WPTR   *temp, *temp1;
  printf("maximum depth ? ");
```

```
scanf("%d", &max_depth);
printf("braid index ? ");
scanf("%d", &bl);
printf("no output (0) or Young tableau (1) and W-graph (2) ? ");
scanf("%d", &g_WI);
for (i = 0; i < size_tableau; i++) {
  tableau[i] = (char *)malloc((long)sizeof(char)*size_tableau);
  if (tableau[i] == NULL) {
    for (j = 0; j < i; j++) free(tableau[j]);
    printf("Memory exhausted in [tableau].¥n");
    exit(0);
  }
}
sprintf(buf, "Y%03d", bl);
if ((g_fp_Y = fopen(buf, "w")) == NULL) {
  printf("File %s can not open.¥n", buf);
  exit(0);
}
real_max_matrix_size = 1;
partition(bl);
for (i = 0; i < size_tableau; i++) free(tableau[i]);
for (i = 0; i < max_matrix_size; i++) incident_matrix[i] = NULL;
sprintf(buf, "RR%03d", bl);
if ((fp_w = fopen(buf, "w")) == NULL) {
  printf("File %s can not open.¥n", buf);
  exit(0);
}
fprintf(fp_w, "%d¥n", g_num_of_rep);
for (k = 0; k < g_num_of_rep; k++) {
  if (g_matrix_size[k] > 1) {
    for (i = 0; i < g_matrix_size[k]-1; i++)
      for (j = i+1; j < g_matrix_size[k]; j++) make_W_graph1(k, i, j);
    p1 = 1;
    do {
      p0 = p1;
      for (i = 0; i < g_matrix_size[k]-1; i++) {
        for (j = i+1; j < g_matrix_size[k]; j++) {
          if (check_W_graph_entry(i, j, p0)) {
            if (p0 == p1) p1++;
            make_W_graph2(k, p0, i, j);
          }
        }
      }
      if (p1 >= max_index) {
        printf("Index overflow.¥n");
        release_memory();
        exit(0);
```

```
        }
      } while (p0 < p1);
      dim = 0;
      reduction_space(k, g_matrix_size[k]);
      write_W_matrix(k);
      fprintf(fp_w, "%1d %1d ", g_matrix_size[k], dim);
      for (i = 0; i < dim; i++) fprintf(fp_w, "%1d ", g_red_entry[i]);
      fprintf(fp_w, "¥n");
      make_representation(k, g_matrix_size[k]);
      for (m = 0; m < g_matrix_size[k]; m++) {
        if (incident_matrix[m] != NULL) {
          temp = incident_matrix[m];
          while (temp != NULL) {
            temp1 = temp->next;
            free(temp);
            temp = temp1;
          }
          incident_matrix[m] = NULL;
        }
      }
    }
  }
  fprintf(fp_w, "¥n"); fclose(fp_w); fclose(g_fp_Y);
  printf("All have completed.¥n");
  release_memory();
}
```

結び目理論研究支援ソフトウエア KnotTheorybyComputer

KNOTTHEORYbyCOMPUTER is a computer software for Apple Macintosh computers being developed in order to assist researchers in knot theory. This program

(1) draws a picture of a knot or a link on the display from a given P-data.
(2) allows the user to draw a picture of a knot by the mouse for inputting a knot.
(3) simplifies a given diagram of a link, and determines whether the knot represented by a diagram of about 50 crossings or less is trivial or not.
(4) computes polynomial invariants of links such as those of Alexander, Conway, Jones, HOMFLY, Q, and Kauffman.
(5) allows the user to draw a picture of a braid, and compute the Alexander and the Jones polynomial of the closed braid, using Matrix representations of the braid (up to 10 string braids).
(6) allows the user to apply Reidemeister, Markov, and other moves to a shown diagram by clicking the mouse.
(7) allows the user to decompose a knot into a tangle, and execute 3 kinds of mutations of the tangle.
(8) allows the user to deform the link by edge-moves or vertex-moves using tracking mouse in the PL-drawing mode.
(9) computes 3-parallel polynomial invariants of 3 and 4 braids which can recognize Kinoshita-Terasaka and Conway knots.

The program has many more features (some of which have not been documented).

Getting started.

Start the program by double-clicking on its icon. A blank window will appear, and the menu bar will show 6 items: File, Edit, DRAW, ACTION, DEFORM, and MODE&UTIL.

The first two of them are used for manipulating text files only. You can use them to prepare P-data files or to edit log files. Note that the name of a P-data file must end with the extension '.prd', as in 'trivial.prd'. The program supports HFS file system, but all the log files are created in the folder which contains the application.

The third menu DRAW is used for inputting a knot using the mouse. Choose 'Draw a Knot'. Click the mouse to start drawing a knot. Move the mouse at a moderate speed while drawing. When the mouse pointer comes back to the starting point, the program converts the picture to a diagram with a small circle over each crossing. You can then change any of the crossings by clicking over the small circle. When you click somewhere else, the program creates the P-data for the knot and outputs it in the current log file. If no log file is opened, the P-data is written in the file named 'WORK.LOG'. The current version of

KNOTTHEORYbyCOMPUTER does not support inputting a link by this method.

You can alternatively draw a picture of a knot or a link by 'PL Draw a link'. By choosing this menu, one enters the PL draw mode. One can then draw a diagram of a knot or a link "piecewise linearly" by clicking vertices. One can cancel drawing the last edge by pressing the "Delete" key. Note that one remains in the PL draw mode until one chooses the "..Exit" from the "DRAW" menu. Once you have constructed a knot and link, you may modify it by clicking any vertex and then by pressing the "Delete" key. The selected vertex becomes the starting vertex of the current PL-data. Furthermore, you can save and load the PL-data by the corresponding menus appearing when you enter into the PL draw mode.

This mode also contains a facility to compute rapidly the one variable Alexander polynomial of a link with a great many crossing points using the Alexander matrix of the link. The routine makes use of the fraction free Gaussian elimination method to compute determinants of Alexander matrixes and so has been implemented multiple precision integers to compute Alexander polynomials of large links to avoid coefficient expansions.

Moreover, you can draw a picture of an n-string braid by 'Draw a braid' . Entering this menu, you input a braid as follows:

abcdefghABCDEFGHb^5

where letters express generators of Braid groups, capital letters do inverses of them, and the symbol ^ gives a power of generators.

You can also input three commands 'open', 'read', and 'close' to substitute to input a word (see Moody.data in the disk). Note that in a data file, one data must be terminated at a carrige return. We will improve the user interface of this routine in the next version.

The ACTION menu.

The main menu is ACTION. It contains 13 submenus. You choose 'OPEN a data file' submenu to open a P-data file. 'READ a P-data' reads one P-data. Choosing 'COMPUTE invariants' computes polynomial invariants of links using the Conway relation and writes them in the current log file. By default the program computes the Jones polynomial only; you can change it by the 'SET invariants' submenu in the 'MODE&UTIL' menu. Although all the computed invariants are outputted in the log file, only the default invariant is displayed on the window within a certain screen limit size; scrolling the window displaying the invariant is not supported in the current version of the program.

After reading a P-data, you can see a diagram of the knot by choosing 'SHOW a link' submenu. By choosing 'SELECT a domain' in the DEFORM menu, one can specify the outer region for drawing the diagram.

'JUDGE a knot to be trivial' in the ACTION menu determines if the given knot is a trivial knot by searching for up to 2-waves. If no 2-waves are found, the program applies Reidemeister moves of types II and III under a certain rule to find 2-waves. The judgment is correct for almost all diagrams of about 50 crossings or less, although some diagrams of the trivial knot for which the program fails to judge have been found (see Figure 1.0).

About P-data.

KNOTTHEORYbyCOMPUTER uses a data structure called P-data to internally represent a knot or a link. A P-data is defined as follows: Given a diagram of a knot or a link, trace a component starting from an arbitrary point and number each crossing one encounters (1, ..., m_1) until one returns to the starting point. For a link diagram, trace another component and continue numbering crossings (m_1+1, ...). When the process is done, each crossing should be assigned two numbers, one for the undercrossing arc and the other for the overcrossing arc. The P-data is

2n c

m_1 m_2 ... m_c

a_1 a_2 ... a_n

where n is the number of crossings, c is the number of components, m_i is the number of crossings on the i-th component. When one traces the components of the diagram in the same way as he/she numbered crossings, at the i-th overcrossing point, the number attached to the corresponding undercrossing point is the absolute value of a_i. The sign of a_i represents the algebraic crossing number at the crossing.

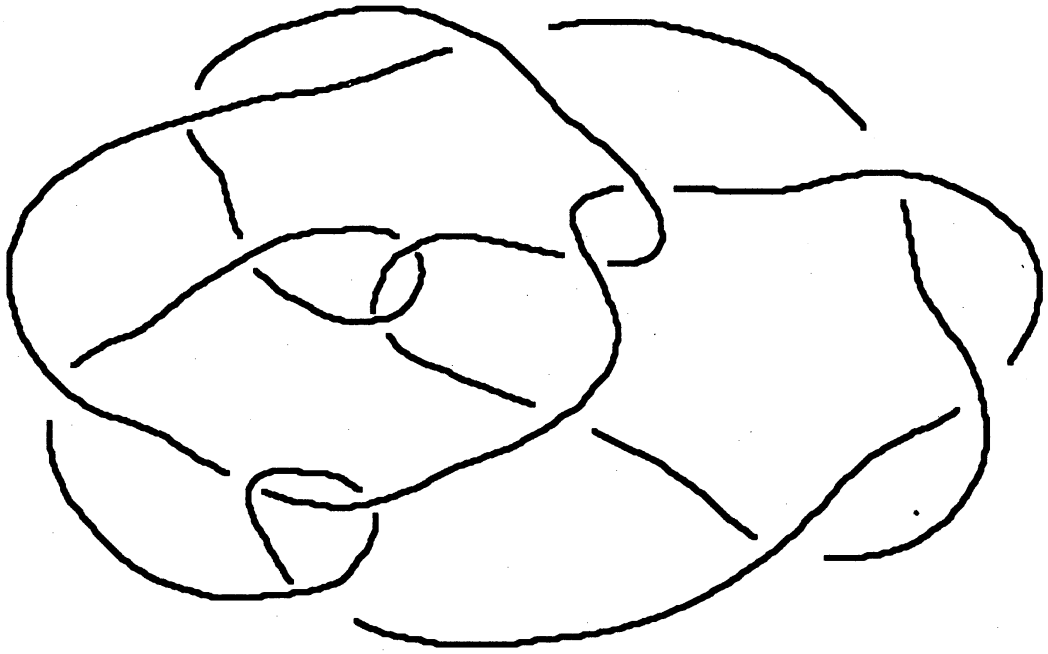


Figure 1.0

For instance, the above diagram of the trivial knot is drawn by the program from the following P-data

```
30  1
30
11  17 -14 -27  25 -20   1  23   3 -10  15  21 -29  -7  13
 555
```

(The 555 in the last line indicates the end of the P-data).

Comments and inquiries are welcomed, and send them to

Mitsuyuki Ochiai
Department of Information and Computer Sciences
Nara Women's University
Kita-Uoya Nishimachi, Nara 630 JAPAN
email: ochiai@ics.nara-wu.ac.jp

P.S. I have planning to improve the user interface of the next version greatly and the improved version will be distributed only by Network, where the network address is wuarchive.wustl.edu. Please contact professor Earl D. Fife, if you have an any interest. If you are not available for E-mail, please inform me of your hope by a real-mail.

Earl D. Fife
Professor of Mathematics
Calvin College
email: fife@calvin.edu